WEST Search History

Hide Items | Restore | Clear | Cancel

DATE: Thursday, November 02, 2006

Hide?	Set Name	Query	Hit Count
	DB=PGPB; PLUR=YES; OP=ADJ		
☐	L39	(vector and bit$1 and sweep and mark$3 and garbage and colect$3).clm.	0
☐	L38	(just-in-time and compil$3 and sweep).clm.	2
☐	L37	(virtual and machine and mark and bit and vector and point$3 and garbage).clm.	3
☐	L36	(virtual and machine and mark and bit and vector and point$3).clm.	3
☐	L35	(mark and sweep and garbage and collect$3 and root and enumerat$4 and live and object$1 and virtual and machine).clm.	2
☐	L34	(mark and sweep and garbage and collect$3 and root and enumerat$4 and live and object$1 and search).clm.	2
☐	L33	(mark and sweep and garbage and collect$3 and root and enumerat$4 and live and object$1 and toggle).clm.	1
☐	L32	(garbage and collect$3 and live and object$1 and heap and block$1 and bit and storage and space).clm.	2
☐	L31	(garbage and collect$3 and live and object$1 and heap and block$1 and bit).clm.	2
☐	L30	(garbage and collect$3 and live and object$1 and heap and block$1).clm.	3
☐	L29	(garbage and collect$3 and live and object$1).clm.	28
☐	L28	(mark and sweep and garbage and collect$3 and root and enumerat$4 and live and object$1).clm.	2
☐	L27	(mark and sweep and garbage and collect$3 and root and enumerat$4).clm.	2
☐	L26	(mark and sweep and garbage and collect$3).clm.	12
☐	L25	(memory and space and garbage and collect$3 and invok$3 and heap).clm.	2
☐	L24	(memory and space and garbage and collect$3 and invok$3 and concurrent).clm.	0
☐	L23	(memory and space and garbage and collect$3 and invok$3 and concurrent$3).clm.	0
☐	L22	(memory and space and garbage and collect$3 and invok$3).clm.	6
☐	L21	(memory and space and garbage and collect$3 and threshold$3 and vector and heap).clm.	0
☐	L20	(memory and space and garbage and collect$3 and threshold$3).clm.	9
☐	L19	(memory and space and garbage and collect$3).clm.	67
☐	L18	(garbage and collect$4 and toggl$3 and mark$3).clm.	1
☐	L17	(heap and vector and second and bit and mark and sweep).clm.	2

☐	L16	(heap and vector and second and bit).clm.	6
☐	L15	(memory and space and vector and pointer$1 and bit and sweep).clm.	1
☐	L14	(mark-sweep and mark$3 and phase and bit and vector$1).clm.	1
☐	L13	(first and second and vector$1 and bit$1 and garbage and thread and trac$3).clm.	2
☐	L12	(first and second and vector$1 and bit$1 and garbage and threshold).clm.	3
☐	L11	(first and second and vector$1 and bit$1 and garbage and live and object$1).clm.	2
☐	L10	(first and second and vector$1 and bit$1).clm.	1175
☐	L9	(mark and sweep and software and concurrent$3).clm.	2
☐	L8	(mark and sweep and garbage and collect$4 and heap and (memory or space) and bit and vector$1 and pointer$1 and threshold$3 and thread$1 and block$1).clm.	2
☐	L7	(mark and sweep and garbage and collect$4 and heap and (memory or space) and bit and vector$1 and pointer$1 and threshold$3 and thread$1).clm.	2
☐	L6	(mark and sweep and garbage and collect$4 and heap and (memory or space) and bit and vector$1 and pointer$1 and threshold$3).clm.	2
☐	L5	(mark and sweep and garbage and collect$4 and heap and (memory or space) and bit and vector$1 and pointer$1).clm.	2
☐	L4	(mark and sweep and garbage and collect$4 and heap and (memory or space) and bit and vector$1).clm.	2
☐	L3	(mark and sweep and garbage and collect$4 and heap and (memory or space)).clm.	7
☐	L2	(mark and sweep and garbage and collect$4 and heap).clm.	7
☐	L1	(mark and sweep and garbage and collect$4).clm.	12

END OF SEARCH HISTORY

WEST Search History

Hide Items | Restore | Clear | Cancel

DATE: Thursday, November 02, 2006

Hide?	Set Name	Query	Hit Count
		DB=PGPB,USPT,USOC,EPAB,JPAB,DWPI,TDBD; PLUR=YES; OP=ADJ	
☐	L162	L160 and vector and pointer	0
☐	L161	L160 and (bit near5 vector)	0
☐	L160	L159 and 'just-in-time'	9
☐	L159	6529919 .uref.	12
☐	L158	L157 and (storage near5 space)	3
☐	L157	L147 and (vector near5 pointer$1)	18
☐	L156	L155 and (sweep near5 bit)	0
☐	L155	L152 and vector$1 and bit$1 and pointer$1	3
☐	L154	L152 and toggl$4	0
☐	L153	L152 and toggle	0
☐	L152	L151 and 'just-in-time'	13
☐	L151	L150 and mark$3 and sweep$3 and heap	15
☐	L150	L149 and (native near5 code)	59
☐	L149	L148 and (virtual near5 machine)	177
☐	L148	L147 and compil$3	213
☐	L147	l130 and java	297
☐	L146	L145 and sweep$3 and storage and space and aloca$5	1
☐	L145	L144 and heap	30
☐	L144	l130 and root and execut$3 and bit and vector$1	47
☐	L143	L142 and memory and space	3
☐	L142	L141 and sweep$3 and mark$3 and heap	3
☐	L141	L140 and thread and trac$3	10
☐	L140	l130 and (vector near5 pointer$1)	25
☐	L139	L138 and bit and pointer$1	1
☐	L138	L137 and vector$1 and toggl$3	1
☐	L137	l134 and (garbage near5 collect$3)	16
☐	L136	L135 and mark and sweep	0
☐	L135	L134 and (heap same threshold$3)	2
☐	L134	6321240 .uref.	17
☐	L133	L132 and (bit near5 vector$1)	7

☐	L132	L131 and (live near5 object$1)	96
☐	L131	L130 and thread	228
☐	L130	L124 and (garbage near5 collect$3) and @py<=2003	580
☐	L129	L124 and (garbage near5 collect$3)	1142
☐	L128	L127 and (heap near5 block$1) and @py<=2003	2
☐	L127	L126 and (concurrent$3 near5 collect$3)	62
☐	L126	L125 and (garbage near5 collect$4)	207
☐	L125	L124 and (mark$3 near5 sweep$3)	212
☐	L124	L123 or l122 or l121	20584
☐	L123	717/114,116,142,148,165.ccls.	1361
☐	L122	711/133,170-173,159-160,165.ccls.	5751
☐	L121	707/1,100,103R-103Y,200,206.ccls.	13837
☐	L120	L119 and heap and garbage	6
☐	L119	(sweep near5 bit) and (mark near5 bit) and @py<=2004	31
☐	L118	L117 and initializ$3	7
☐	L117	L116 and bit and vector	11
☐	L116	L114 and header and field$1 and pointer$1	40
☐	L115	L114 and (data near5 structurre)	0
☐	L114	L113 and (mark near5 sweep)	122
☐	L113	(heap and block$1 and garbage and collection) and @py<=2004	591
☐	L112	(heap and block$1 and garbage and collection).ti. and @py<=2004	0
☐	L111	L107 and (garbage near5 collection)	2
☐	L110	L107 and mark and sweep	0
☐	L109	L107 and mark and sweep and (memory near5 structure)	0
☐	L108	L107 and mark and sweep and (data near5 structure)	0
☐	L107	(header same (heap near5 block$1)) and @py<=2004	25
☐	L106	(mark and sweep and heap).ti. and @py<=2004	1
☐	L105	(mark and sweep and structure).ti. and @py<=2004	0
☐	L104	L103	0
☐	L103	(mark and sweep and garbage and structure).ti. and @py<=2004	0
☐	L102	(mark and sweep and field$1 and memory and structure).ab. and @py<=2004	0
☐	L101	(mark and sweep and field$1 and memory and structure).ti. and @py<=2004	0
☐	L100	(data and structure and heap).ti. and @py<=2004	13
☐	L99	'data strcucture'.ti.	0
☐	L98	'data strcucture'.ti. and @py<=2004	0
☐	L97	L96 and (bit near5 vector$1)	6
☐	L96	(memory near5 structure) and (mark$sweep) and field$1 and pointer$1 and @py<=2004	66

☐	L95	(mark near5 field$1) and (sweep near5 field$1) and (data near5 structure) and @py<=2004	6
☐	L94	L93 and vector$1 and heap	6
☐	L93	L92 and field$1 and status	8
☐	L92	L91 and bit$1 and object$1	17
☐	L91	L90 and pointer$1	18
☐	L90	L89 and (mark near5 sweep)	18
☐	L89	4907151.uref.	66
☐	L88	L87 and field$1 and header	1
☐	L87	L86 and (mark near5 sweep)	15
☐	L86	L85 and (heap near5 block$1)	107
☐	L85	(heap near5 structure) and @py<=2004	669
☐	L84	L83 and (header near5 field$1)	2
☐	L83	L82 and (mark near5 bit$1)	13
☐	L82	L81 and (heap near5 block$1)	100
☐	L81	(garbage near5 collection) and (data near5 structure) and @py<=2004	1495
☐	L80	L79 and (multiple near5 object$1)	4
☐	L79	L78 and (multiple near5 field$1)	39
☐	L78	L77 and pointer$1	383
☐	L77	(heap near5 structure) and @py<=2004	669
☐	L76	L75 and ((second near5 field) same (sweep near5 bit))	1
☐	L75	(first near5 field) same (mark near5 bit)	96
☐	L74	L73 and vector	4
☐	L73	L72 and (field$1 near5 object$1)	22
☐	L72	L71 and status	23
☐	L71	L69 and heap	28
☐	L70	L69 and toggl$2	0
☐	L69	L68 and (mark near5 sweep)	39
☐	L68	(data near5 structure) and (memory near5 structure) and header and @py<=2004	6667
☐	L67	L66 and garbage	4
☐	L66	L65 and pointer$1	23
☐	L65	L62 and (mark near5 bit$1)	23
☐	L64	L62 and (mark near5 bit$1) and (sweep near5 bit$1)	0
☐	L63	L62 and (mark near5 field$1) and (sweep near5 field$1)	1
☐	L62	(heap near5 structure) and (memory near5 structure) and header and @py<=2004	125
☐	L61	(data near5 structure) and (memory near5 structure) and (garbage near5 collection) and heap and mark and sweep and bit and vector and field$1 and	0

		statuses and object$1 and @py<=2004	
☐	L60	L59 and ((sweep near5 bit) same pointer)	0
☐	L59	L49 and ((mark near5 bit) same pointer$1)	5
☐	L58	(mark near5 field$1) and (sweep near5 field$1) and heap and garbage and collection and algorithm$1 and statu$3 and @py<=2004	2
☐	L57	L55 and field$1 and status	5
☐	L56	L55 and (field$1 near5 status)	0
☐	L55	L54 and (garbage near5 collection$1)	7
☐	L54	L52 and field$1	22
☐	L53	L52 and (heap near5 block$1)	2
☐	L52	(mark near5 bit$1) and (sweep near5 bit$1) and @py<=2004	31
☐	L51	(mark near5 bit$1) and (sweep near5 bit$1) and (sweep$3 near5 status$2) and (memory near5 field$1) and @py<=2004	0
☐	L50	L49 and (mark near5 bit$1) and (sweep near5 bit$1)	2
☐	L49	(memory near5 structure) and (heap near5 block$1) and @py<=2004	187
☐	L48	L47 and (live near5 object$1)	12
☐	L47	L46 and scann$3	17
☐	L46	L45 and pointer$1	28
☐	L45	L44 and (heap near5 block)	30
☐	L44	(garbage near5 sweep$3) and @py<=2004	212
☐	L43	(garbage near5 sweep$3) and (sweep nar5 bit) and @py<=2004	0
☐	L42	(live near5 object$1) scann$3 and search$3 and mark$3 and thread$1	5
☐	L41	L40 and (live near5 object)	7
☐	L40	L39 and vector	7
☐	L39	L38 and (mark near5 sweep)	10
☐	L38	(garbage near5 sweep$3) and (sweep near5 storage) and @py<=2004	10
☐	L37	(garbage near5 sweep$3) and (sweep near5 storage) and (stroage near5 space) and @py<=2004	0
☐	L36	L35 and live	3
☐	L35	L34 and heap	5
☐	L34	L33 and pointer$1	5
☐	L33	L25 and (sweep near5 bit)	5
☐	L32	L29 and vector and bit$1 and mark and sweep	1
☐	L31	L29 and (sweep near5 stroage)	0
☐	L30	L29 and (bit near5 vector$1)	2
☐	L29	(live near5 object$1) and (heap near5 block) and @py<=2004	69
☐	L28	L27 and vector$1 and bit$1	1
☐	L27	L26 and (mark near5 sweep)	12
☐	L26	L25 and (live near5 object$1)	59

☐	L25	(heap near5 block) and (data near5 structure) and @py<=2004	252
☐	L24	L23 and (sweep near5 bit)	2
☐	L23	L22 and (mark near5 bit)	12
☐	L22	(heap near5 block) and (mark near5 sweep) and @py<=2004	29
☐	L21	(mark and sweep and garbage and collection).ti. and @py<=2004	2
☐	L20	L19 and conflict$3	0
☐	L19	L18 and thread$1	6
☐	L18	L17 and scann$3	8
☐	L17	L16 and execut$3	11
☐	L16	L15 and (mark near5 bit)	11
☐	L15	L14 and (live near5 object$1)	13
☐	L14	L13 and (garbage near5 collect$3)	13
☐	L13	L12 and (mark near5 sweep)	13
☐	L12	L11 and vector$1 and bit$1	97
☐	L11	L10 and virtual and java	782
☐	L10	just-in-time and @py<=2004	2677
☐	L9	L8 and trace	1
☐	L8	L7 and root	4
☐	L7	L6 and (mark near5 sweep)	4
☐	L6	L5 and (live near5 object$1)	22
☐	L5	L4 and concurrently	56
☐	L4	vector$1 and bit$1 and pointer$1 and heap and block$1 and live and object$1 and @py<=2004	131
☐	L3	(second near5 bit) same (vector near5 pointer$1) and heap and @py<=2004	0
☐	L2	(second near5 bit) same (vector near5 pointer$1) and heap and memory and @py<=2004	0
☐	L1	(second near5 bit) same (vector near5 pointer$1) and (heap near5 block) and @py<=2004	0

END OF SEARCH HISTORY

10/719,443 STIC/EIC Search 11/2/2006

```
File   9:Business & Industry(R) Jul/1994-2006/Nov 01
         (c) 2006  The Gale Group
File  13:BAMP 2006/Oct W4
         (c) 2006  The Gale Group
File  16:Gale Group PROMT(R) 1990-2006/Nov 02
         (c) 2006 The Gale Group
File  47:Gale Group Magazine DB(TM) 1959-2006/Nov 01
         (c) 2006 The Gale group
File  88:Gale Group Business A.R.T.S. 1976-2006/Nov 01
         (c) 2006 The Gale Group
File 148:Gale Group Trade & Industry DB 1976-2006/Nov 02
         (c)2006 The Gale Group
File 160:Gale Group PROMT(R) 1972-1989
         (c) 1999 The Gale Group
File 275:Gale Group Computer DB(TM) 1983-2006/Nov 02
         (c) 2006 The Gale Group
File 621:Gale Group New Prod.Annou.(R) 1985-2006/Nov 01
         (c) 2006 The Gale Group
File 624:McGraw-Hill Publications 1985-2006/Nov 02
         (c) 2006 McGraw-Hill Co. Inc
File 634:San Jose Mercury  Jun 1985-2006/Oct 31
         (c) 2006 San Jose Mercury News
File 636:Gale Group Newsletter DB(TM) 1987-2006/Nov 02
         (c) 2006 The Gale Group
File 649:Gale Group Newswire ASAP(TM) 2006/Oct 19
         (c) 2006 The Gale Group
File 647:CMP  Computer Fulltext 1988-2006/Dec W4
         (c) 2006 CMP Media, LLC
File 674:Computer News Fulltext 1989-2006/Sep W1
         (c) 2006 IDG Communications

Set     Items   Description
S1        824   (BIT OR BINARY()DIGIT? ?)(1W)VECTOR? ? OR BITVECTOR?
S2       3738   MARK???(1W)(SWEEP? OR COMPACT????) OR MARKSWEEP? OR MARKCO-
            MPACT?
S3        153   HEAP? ?(2N)(BLOCK? ? OR CHUNK? ?) OR HEAPBLOCK? OR HEAPCHU-
            NK?
S4      56876   HEAP? ? OR FREESTORE? OR FREE()STORE? ?
S5    1541771   JAVA OR JVM OR JVMS OR VS OR VM OR VMS OR JRE? ? OR VME? ?
            OR VMM OR VMMS OR VIRTUAL(1W)(PC OR COMPUTER? ?)
S6        302   METAMACHINE? OR (META OR ABSTRACT)()MACHINE? ?
S7         11   PSEUDO()(OS OR OPERATING()SYSTEM? ?)
S8     179913   COREWAR? OR CORE()WAR OR OCODE OR POPLOG OR PORTABLE()SCHE-
            ME()INTERPRET? OR PORTABLE()STANDARD()LISP OR OS2 OR OS()2
S9     134959   SEQUENTIAL()PARLOG()MACHINE? ? OR PYTHON OR SMALLTALK OR S-
            MALL()TALK OR SNOBOL OR IMPLEMENTATION()LANGUAGE? ? OR SODA
S10     10806   VMWARE OR HYPERVIS?R? ? OR XEN
S11     76773   GARBAGE
S12   5080292   MEMORY OR STORAGE OR SPACE
S13    145476   JUST(1W)TIME? ? OR JIT OR KANBAN OR KAN()BAN
S14         0   S1(S)S2
S15      1943   VECTOR? ?(3N)(BIT OR BINARY()DIGIT? ?)
S16         0   S15(S)S2
S17         7   S2(S)S3:S4
S18         3   S17(S)S5:S10
S19         1   S1(S)S11
S20         4   S18:S19
S21         1   S20/2004:2006
S22         3   S20 NOT S21
S23         3   RD  (unique items)

 23/3,K/1     (Item 1 from file: 88)
DIALOG(R)File  88:Gale Group Business A.R.T.S.
(c) 2006 The Gale Group. All rts. reserv.
```

```
File 696:DIALOG Telecom. Newsletters 1995-2006/Nov 01
         (c) 2006 Dialog
File  15:ABI/Inform(R) 1971-2006/Nov 01
         (c) 2006 ProQuest Info&Learning
File  98:General Sci Abs 1984-2006/Oct
         (c) 2006 The HW Wilson Co.
File 112:UBM Industry News 1998-2004/Jan 27
         (c) 2004 United Business Media
File 141:Readers Guide 1983-2006/Aug
         (c) 2006 The HW Wilson Co
File 484:Periodical Abs Plustext 1986-2006/Oct W4
         (c) 2006 ProQuest
File 813:PR Newswire 1987-1999/Apr 30
         (c) 1999 PR Newswire Association Inc
File 613:PR Newswire 1999-2006/Nov 02
         (c) 2006 PR Newswire Association Inc
File 635:Business Dateline(R) 1985-2006/Nov 01
         (c) 2006 ProQuest Info&Learning
File 810:Business Wire 1986-1999/Feb 28
         (c) 1999 Business Wire
File 610:Business Wire 1999-2006/Nov 02
         (c) 2006 Business Wire.
File 369:New Scientist 1994-2006/Aug W4
         (c) 2006 Reed Business Information Ltd.
File 370:Science 1996-1999/Jul W3
         (c) 1999 AAAS

Set     Items   Description
S1        104   (BIT OR BINARY()DIGIT? ?)(1W)VECTOR? ? OR BITVECTOR?
S2        917   MARK???(1W)(SWEEP? OR COMPACT????) OR MARKSWEEP? OR MARKCO-
              MPACT?
S3         26   HEAP? ?(2N)(BLOCK? ? OR CHUNK? ?) OR HEAPBLOCK? OR HEAPCHU-
              NK?
S4      23106   HEAP? ? OR FREESTORE? OR FREE()STORE? ?
S5     391372   JAVA OR JVM OR JVMS OR VS OR VM OR VMS OR JRE? ? OR VME? ?
              OR VMM OR VMMS OR VIRTUAL(1W)(PC OR COMPUTER? ?)
S6        130   METAMACHINE? OR (META OR ABSTRACT)()MACHINE? ?
S7          4   PSEUDO()(OS OR OPERATING()SYSTEM? ?)
S8      32165   COREWAR? OR CORE()WAR OR OCODE OR POPLOG OR PORTABLE()SCHE-
              ME()INTERPRET? OR PORTABLE()STANDARD()LISP OR OS2 OR OS()2
S9      44076   SEQUENTIAL()PARLOG()MACHINE? ? OR PYTHON OR SMALLTALK OR S-
              MALL()TALK OR SNOBOL OR IMPLEMENTATION()LANGUAGE? ? OR SODA
S10      2673   VMWARE OR HYPERVIS?R? ? OR XEN
S11     35458   GARBAGE
S12   1703232   MEMORY OR STORAGE OR SPACE
S13     61046   JUST(1W)TIME? ? OR JIT OR KANBAN OR KAN()BAN
S14         0   S1(S)S2
S15       272   VECTOR? ?(3N)(BIT OR BINARY()DIGIT? ?)
S16         0   S15(S)S2
S17         0   S2(S)S3:S4
S18         0   S1(S)S11
```

File 347:JAPIO Dec 1976-2006/Jan(Updated 061009)
(c) 2006 JPO & JAPIO
File 348:EUROPEAN PATENTS 1978-2006/ 200643
(c) 2006 European Patent Office
File 349:PCT FULLTEXT 1979-2006/UB=20061026UT=20061019
(c) 2006 WIPO/Thomson
File 350:Derwent WPIX 1963-2006/UD=200670
(c) 2006 The Thomson Corporation

Set	Items	Description
S1	18	AU='SUBRAMONEY S'
S2	16	AU='SUBRAMONEY SREENIVAS':AU='SUBRAMONEY SREEVINAS'
S3	174	AU='HUDSON R':AU='HUDSON R W'
S4	52	AU='HUDSON RICHARD':AU='HUDSON RICHARD 1602 BLUEBERRY IMPERIAL MISSOUR'
S5	16	S1:S2 AND S3:S4
S6	1913	BIT(1W)VECTOR? ?
S7	2	S5 AND S6

>>>Format 69 is not valid in file 348

7/69/1 (Item 1 from file: 350)
DIALOG(R)File 350:Derwent WPIX

0015288123 - Drawing available
WPI ACC NO: 2005-638261/200565
XRPX Acc No: N2005-523511
Mark-sweep-compact garbage collection method in managed runtime system, involves using only one bit vector of each heap block for performing garbage collection, if available space in heap is below preset value
Patent Assignee: HUDSON R L (HUDS-I); SUBRAMONEY S (SUBR-I)
Inventor: **HUDSON R L** ; **SUBRAMONEY S**
Patent Family (1 patents, 1 countries)

Patent Number	Kind	Date	Application Number	Kind	Date	Update
US 20050198088	A1	20050908	US 2004793707	A	20040303	200565 B

Priority Applications (no., kind, date): US 2004793707 A 20040303

Patent Details

Number	Kind	Lan	Pg	Dwg	Filing Notes
US 20050198088	A1	EN	29	13	

Alerting Abstract US A1
NOVELTY - The method involves using only one **bit vector** of each of several heap blocks of a heap, for performing mark-sweep-compact garbage collection in the heap, if the available space in the heap is below the threshold value. The heap blocks are marked, compacted or swept by only one **bit vector** of each heap block.
DESCRIPTION - INDEPENDENT CLAIMS are also included for the following:

1.mark-sweep-compact garbage collection performing system;

2.method for automatically collecting garbage objects;

3.managed runtime system;

4.article comprising medium storing mark-sweep-compact garbage collection program; and

5.article comprising medium storing program for automatically collecting garbage objects.

USE - For performing mark-sweep-compact garbage collection in managed

```
File   2:INSPEC 1898-2006/Oct W4
         (c) 2006 Institution of Electrical Engineers
File   6:NTIS 1964-2006/Oct W4
         (c) 2006 NTIS, Intl Cpyrght All Rights Res
File   8:Ei Compendex(R) 1970-2006/Oct W4
         (c) 2006 Elsevier Eng.  Info. Inc.
File  34:SciSearch(R) Cited Ref Sci 1990-2006/Oct W4
         (c) 2006 The Thomson Corp
File  35:Dissertation Abs Online 1861-2006/Oct
         (c) 2006 ProQuest Info&Learning
File  65:Inside Conferences 1993-2006/Nov 01
         (c) 2006 BLDSC all rts. reserv.
File  94:JICST-EPlus 1985-2006/Jul W3
         (c)2006 Japan Science and Tech Corp(JST)
File  95:TEME-Technology & Management 1989-2006/Oct W5
         (c) 2006 FIZ TECHNIK
File  99:Wilson Appl. Sci & Tech Abs 1983-2006/Sep
         (c) 2006 The HW Wilson Co.
File 111:TGG Natl.Newspaper Index(SM) 1979-2006/Oct 19
         (c) 2006 The Gale Group
File 144:Pascal 1973-2006/Oct W2
         (c) 2006 INIST/CNRS
File 256:TecInfoSource 82-2006/Apr
         (c) 2006 Info.Sources Inc
File 266:FEDRIP 2006/Aug
         Comp & dist by NTIS, Intl Copyright All Rights Res
File 434:SciSearch(R) Cited Ref Sci 1974-1989/Dec
         (c) 2006 The Thomson Corp
File 483:Newspaper Abs Daily 1986-2006/Nov 02
         (c) 2006 ProQuest Info&Learning
File 583:Gale Group Globalbase(TM) 1986-2002/Dec 13
         (c) 2002 The Gale Group
File  56:Computer and Information Systems Abstracts 1966-2006/Oct
         (c) 2006 CSA.
File  57:Electronics & Communications Abstracts 1966-2006/Oct
         (c) 2006 CSA.
File  60:ANTE: Abstracts in New Tech & Engineer 1966-2006/Oct
         (c) 2006 CSA.

Set     Items   Description
S1       1311   (BIT OR BINARY()DIGIT? ?)(1W)VECTOR? ? OR BITVECTOR?
S2        693   MARK???(1W)(SWEEP? OR COMPACT????) OR MARKSWEEP? OR MARKCO-
                MPACT?
S3         34   HEAP? ?(2N)(BLOCK? ? OR CHUNK? ?) OR HEAPBLOCK? OR HEAPCHU-
                NK?
S4      13727   HEAP? ? OR FREESTORE? OR FREE()STORE? ?
S5     743031   JAVA OR JVM OR JVMS OR VS OR VM OR VMS OR JRE? ? OR VME? ?
                OR VMM OR VMMS OR VIRTUAL(1W)(PC OR COMPUTER? ?)
S6       4840   METAMACHINE? OR (META OR ABSTRACT)()MACHINE? ?
S7          4   PSEUDO()(OS OR OPERATING()SYSTEM? ?)
S8       7018   COREWAR? OR CORE()WAR OR OCODE OR POPLOG OR PORTABLE()SCHE-
                ME()INTERPRET? OR PORTABLE()STANDARD()LISP OR OS2 OR OS()2
S9      47801   SEQUENTIAL()PARLOG()MACHINE? ? OR PYTHON OR SMALLTALK OR S-
                MALL()TALK OR SNOBOL OR IMPLEMENTATION()LANGUAGE? ? OR SODA
S10       986   VMWARE OR HYPERVIS?R? ? OR XEN
S11     33313   GARBAGE
S12   3988692   MEMORY OR STORAGE OR SPACE
S13     26225   JUST(1W)TIME? ? OR JIT OR KANBAN OR KAN()BAN
S14         2   S1 AND S2
S15      2657   VECTOR? ?(3N)(BIT OR BINARY()DIGIT? ?)
S16         2   S15 AND S2
S17       108   S2 AND S3:S4
S18        69   S17 AND S5:S10
S19         6   S1 AND S11
S20         6   S14 OR S16 OR S19
```

S21 0 S20/2004:2006
S22 4 RD S20 (unique items)
S23 9 S18 AND (BIT OR BITS OR VECTOR? ?)
S24 0 S23/2004:2006
S25 9 S23 NOT S20
S26 5 RD (unique items)

File 348:EUROPEAN PATENTS 1978-2006/ 200643
(c) 2006 European Patent Office
File 349:PCT FULLTEXT 1979-2006/UB=20061026UT=20061019
(c) 2006 WIPO/Thomson

Set	Items	Description
S1	1591	(BIT OR BINARY()DIGIT? ?)(1W)VECTOR? ? OR BITVECTOR?
S2	247	MARK???(1W)(SWEEP? OR COMPACT????) OR MARKSWEEP? OR MARKCOMPACT?
S3	57	HEAP? ?(2N)(BLOCK? ? OR CHUNK? ?) OR HEAPBLOCK? OR HEAPCHUNK?
S4	3091	HEAP? ? OR FREESTORE? OR FREE()STORE? ?
S5	125311	JAVA OR JVM OR JVMS OR VS OR VM OR VMS OR JRE? ? OR VME? ? OR VMM OR VMMS OR VIRTUAL(1W)(PC OR COMPUTER? ?)
S6	105	METAMACHINE? OR (META OR ABSTRACT)()MACHINE? ?
S7	7	PSEUDO()(OS OR OPERATING()SYSTEM? ?)
S8	8894	COREWAR? OR CORE()WAR OR OCODE OR POPLOG OR PORTABLE()SCHEME()INTERPRET? OR PORTABLE()STANDARD()LISP OR OS2 OR OS()2
S9	23707	SEQUENTIAL()PARLOG()MACHINE? ? OR PYTHON OR SMALLTALK OR SMALL()TALK OR SNOBOL OR IMPLEMENTATION()LANGUAGE? ? OR SODA
S10	748	VMWARE OR HYPERVIS?R? ? OR XEN
S11	5071	GARBAGE
S12	923325	MEMORY OR STORAGE OR SPACE
S13	3592	JUST(1W)TIME? ? OR JIT OR KANBAN OR KAN()BAN
S14	0	S1(50N)S2
S15	2865	VECTOR? ?(3N)(BIT OR BINARY()DIGIT? ?)
S16	0	S15(50N)S2
S17	39	S2(50N)S3:S4
S18	7	S17(50N)S5:S10
S19	3	S18 AND AC=US/PR AND AY=(1963:2003)/PR
S20	3	S18 AND AC=US AND AY=1963:2003
S21	3	S18 AND AC=US AND AY=(1963:2003)/PR
S22	6	S18 AND PY=1963:2003
S23	7	S19:S22
S24	7	IDPAT (sorted in duplicate/non-duplicate order)
S25	7	IDPAT (primary/non-duplicate records only)
S26	7	S1(25N)(S11 OR GC)
S27	7	S26 NOT S25
S28	4	S27 AND AC=US/PR AND AY=(1963:2003)/PR
S29	4	S27 AND AC=US AND AY=1963:2003
S30	4	S27 AND AC=US AND AY=(1963:2003)/PR
S31	6	S27 AND PY=1963:2003
S32	6	S28:S31

File 347:JAPIO Dec 1976-2006/Jan(Updated 061009)
(c) 2006 JPO & JAPIO
File 350:Derwent WPIX 1963-2006/UD=200670
(c) 2006 The Thomson Corporation

Set	Items	Description
S1	342	(BIT OR BINARY()DIGIT? ?)(1W)VECTOR? ? OR BITVECTOR?
S2	83	MARK???(1W)(SWEEP? OR COMPACT????) OR MARKSWEEP? OR MARKCOMPACT?
S3	32	HEAP? ?(2N)(BLOCK? ? OR CHUNK? ?) OR HEAPBLOCK? OR HEAPCHUNK?
S4	4444	HEAP? ? OR FREESTORE? OR FREE()STORE? ?
S5	29552	JAVA OR JVM OR JVMS OR VS OR VM OR VMS OR JRE? ? OR VME? ? OR VMM OR VMMS OR VIRTUAL(1W)(PC OR COMPUTER? ?)
S6	27	METAMACHINE? OR (META OR ABSTRACT)()MACHINE? ?
S7	3	PSEUDO()(OS OR OPERATING()SYSTEM? ?)
S8	671	COREWAR? OR CORE()WAR OR OCODE OR POPLOG OR PORTABLE()SCHEME()INTERPRET? OR PORTABLE()STANDARD()LISP OR OS2 OR OS()2
S9	24645	SEQUENTIAL()PARLOG()MACHINE? ? OR PYTHON OR SMALLTALK OR SMALL()TALK OR SNOBOL OR IMPLEMENTATION()LANGUAGE? ? OR SODA
S10	189	VMWARE OR HYPERVIS?R? ? OR XEN
S11	23389	GARBAGE
S12	2975691	MEMORY OR STORAGE OR SPACE
S13	967	JUST(1W)TIME? ? OR JIT OR KANBAN OR KAN()BAN
S14	2	S1 AND S2
S15	879	VECTOR? ?(3N)(BIT OR BINARY()DIGIT? ?)
S16	2	S15 AND S2
S17	0	S16 NOT S14
S18	13	S2 AND S3:S4
S19	1	S18 AND S5:S10
S20	1	S19 NOT S14
S21	6	S1 AND S11
S22	4	S21 AND S3:S4
S23	2	S22 NOT (S14 OR S20)
S24	0	S23 AND AC=US/PR AND AY=(1963:2003)/PR
S25	0	S23 AND AC=US AND AY=1963:2003
S26	0	S23 AND AC=US AND AY=(1963:2003)/PR
S27	1	S23 AND PY=1963:2003
S28	2	S1 AND GC
S29	1	S28 NOT (S20 OR S14 OR S22)

Terms used **mark sweep garbage collection vector**

Found **41,548** of **186,958**

Sort results by: relevance

Display results: expanded form

Save results to a Binder

Search Tips

Open results in a new window

Try an Advanced Search

Try this search in The ACM Guide

Results 1 - 20 of 200 Result page: **1** 2 3 4 5 6 7 8 9 10 next

Best 200 shown

Relevance scale

1 Comparing mark-and sweep and stop-and-copy garbage collection

Benjamin Zorn

May 1990 **Proceedings of the 1990 ACM conference on LISP and functional programming**

Publisher: ACM Press

Full text available: pdf(1.02 MB) Additional Information: full citation, abstract, references, citings, index terms

Stop-and-copy garbage collection has been preferred to mark-and-sweep collection in the last decade because its collection time is proportional to the size of reachable data and not to the memory size. This paper compares the CPU overhead and the memory requirements of the two collection algorithms extended with generations, and finds that mark-and-sweep collection requires at most a small amount of additional CPU overhead (3-6%) but, requires an average of 20% (and up to 40%) less memory t ...

2 A scalable mark-sweep garbage collector on large-scale shared-memory machines

Toshio Endo, Kenjiro Taura, Akinori Yonezawa

November 1997 **Proceedings of the 1997 ACM/IEEE conference on Supercomputing (CDROM)**

Publisher: ACM Press

Full text available: pdf(96.62 KB) Additional Information: full citation, abstract, references, citings

This work describes implementation of a mark-sweep garbage collector (GC) for shared-memory machines and reports its performance. It is a simple "parallel" collector in which all processors cooperatively traverse objects in the global shared heap. The collector stops the application program during a collection and assumes a uniform access cost to all locations in the shared heap. Implementation is based on the Boehm-Demers-Weiser conservative GC (Boehm GC). Experiments have been done on Ultra ...

Keywords: dynamic load balancing, garbage collection, parallel algorithm, scalability, shared-memory machine

3 The derivation of distributed termination detection algorithms from garbage collection schemes

Gerard Tel, Friedemann Mattern

January 1993 **ACM Transactions on Programming Languages and Systems (TOPLAS)**, Volume 15 Issue 1

Publisher: ACM Press

Full text available: pdf(2.36 MB) Additional Information: full citation, abstract, references, citings, index terms

It is shown that the termination detection problem for distributed computations can be modeled as an instance of the garbage collection problem. Consequently, algorithms for the termination detection problem are obtained by applying transformations to garbage collection algorithms. The transformation can be applied to collectors of the "mark-and-sweep" type as well as to reference-counting protocol of Lermen and Maurer, the weighted-reference-counting protocol, the local-referen ...

Keywords: distributed algorithms, distributed termination detection, garbage collection, program transformations

4 A mark-and-sweep collector C++

Daniel R. Edelson
February 1992 **Proceedings of the 19th ACM SIGPLAN-SIGACT symposium on Principles of programming languages**
Publisher: ACM Press

Full text available: pdf(836.06 KB) Additional Information: full citation, abstract, references, citings, index terms

Our research is concerned with compiler-independent, tag-free garbage collection for the C++ programming language. We have previously presented a copying collector based on root registration. This paper presents a mark-and-sweep garbage collector that ameliorates shortcomings of the previous collector. We describe the two collectors and discuss why the new one is an improvement over the old one. We have tested this collector and a conservative collector in a VLSI CAD application, and this p ...

5 Creating and preserving locality of java applications at allocation and garbage collection times

Yefim Shuf, Manish Gupta, Hubertus Franke, Andrew Appel, Jaswinder Pal Singh
November 2002 **ACM SIGPLAN Notices , Proceedings of the 17th ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications OOPSLA '02**, Volume 37 Issue 11
Publisher: ACM Press

Full text available: pdf(180.20 KB) Additional Information: full citation, abstract, references, citings, index terms

The growing gap between processor and memory speeds is motivating the need for optimization strategies that improve data locality. A major challenge is to devise techniques suitable for pointer-intensive applications. This paper presents two techniques aimed at improving the memory behavior of pointer-intensive applications with dynamic memory allocation, such as those written in Java. First, we present an allocation time object placement technique based on the recently introduced notion of *p* ...

Keywords: *JVM, Java, garbage collection, heap traversal, locality, locality based graph traversal, memory allocation, memory management, object co-allocation, object placement, prolific types, run-time systems*

6 Implementing an on-the-fly garbage collector for Java

Tamar Domani, Elliot K. Kolodner, Ethan Lewis, Eliot E. Salant, Katherine Barabash, Itai Lahan, Yossi Levanoni, Erez Petrank, Igor Yanorer
October 2000 **ACM SIGPLAN Notices , Proceedings of the 2nd international symposium on Memory management ISMM '00**, Volume 36 Issue 1
Publisher: ACM Press

Full text available: pdf(1.33 MB) Additional Information: full citation, abstract, citings, index terms

Java uses garbage collection (GC) for the automatic reclamation of computer memory no longer required by a running application. GC implementations for Java Virtual Machines (JVM) are typically designed for single processor machines, and do not necessarily perform well for a server program with many threads running on a multiprocessor. We designed and implemented an on-the-fly GC, based on the algorithm of Doligez, Leroy and Gonthier [13, 12] (DLG), for Java in this environment. An *on-the-f ...*

Keywords: *Java, concurrent garbage collection, garbage collection, memory management, on-the-fly garbage collection, programming languages*

7 MC²: high-performance garbage collection for memory-constrained environments

Narendran Sachindran, J. Eliot B. Moss, Emery D. Berger

October 2004 **ACM SIGPLAN Notices , Proceedings of the 19th annual ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications OOPSLA '04**, Volume 39 Issue 10

Publisher: ACM Press

Full text available: pdf(503.53 KB) Additional Information: full citation, abstract, references, citings, index terms

Java is becoming an important platform for memory-constrained consumer devices such as PDAs and cellular phones, because it provides safety and portability. Since Java uses garbage collection, efficient garbage collectors that run in constrained memory are essential. Typical collection techniques used on these devices are mark-sweep and mark-compact. Mark-sweep collectors can provide good throughput and pause times but suffer from fragmentation. Mark-compact collectors prevent fragmentation, ...

Keywords: copying collector, generational collector, java, mark-compact, mark-copy, mark-sweep, memory-constrained copying

8 Garbage collection and task deletion in distributed applicative processing systems

Paul Hudak, Robert M. Keller

August 1982 **Proceedings of the 1982 ACM symposium on LISP and functional programming**

Publisher: ACM Press

Full text available: pdf(949.06 KB) Additional Information: full citation, abstract, references, citings, index terms

The problem of automatic storage reclamation for distributed implementations of applicative languages is explored. Highly parallel distributed systems have several unique characteristics that complicate the reclamation process; in this setting, the deficiencies of existing storage reclamation schemes are thus noted. A real-time, effectively distributed, garbage collector of the mark-sweep variety, called the marking-tree collector, is shown to accomplish reclamation in parallel ...

9 Garbage collecting the Internet: a survey of distributed garbage collection

Saleh E. Abdullahi, Graem A. Ringwood

September 1998 **ACM Computing Surveys (CSUR)**, Volume 30 Issue 3

Publisher: ACM Press

Full text available: pdf(337.65 KB) Additional Information: full citation, abstract, references, citings, index terms, review

Internet programming languages such as Java present new challenges to garbage-collection design. The spectrum of garbage-collection schema for linked structures distributed over a network are reviewed here. Distributed garbage collectors are classified

first because they evolved from single-address-space collectors. This taxonomy is used as a framework to explore distribution issues: locality of action, communication overhead and indeterministic communication latency.

Keywords: automatic storage reclamation, distributed, distributed file systems, distributed memories, distributed object-oriented management, memory management, network communication, object-oriented databases, reference counting

10 A parallel, incremental, mostly concurrent garbage collector for servers

Katherine Barabash, Ori Ben-Yitzhak, Irit Goft, Elliot K. Kolodner, Victor Leikehman, Yoav Ossia, Avi Owshanko, Erez Petrank

November 2005 **ACM Transactions on Programming Languages and Systems (TOPLAS)**, Volume 27 Issue 6

Publisher: ACM Press

Full text available: pdf(683.50 KB) Additional Information: full citation, abstract, references, index terms

Multithreaded applications with multigigabyte heaps running on modern servers provide new challenges for garbage collection (GC). The challenges for "server-oriented" GC include: ensuring short pause times on a multigigabyte heap while minimizing throughput penalty, good scaling on multiprocessor hardware, and keeping the number of expensive multicycle fence instructions required by weak ordering to a minimum.We designed and implemented a collector facing these demands building on th ...

Keywords: Garbage collection, JVM, concurrent garbage collection

11 Myths and realities: the performance impact of garbage collection

Stephen M. Blackburn, Perry Cheng, Kathryn S. McKinley

June 2004 **ACM SIGMETRICS Performance Evaluation Review , Proceedings of the joint international conference on Measurement and modeling of computer systems SIGMETRICS '04/Performance '04**, Volume 32 Issue 1

Publisher: ACM Press

Full text available: pdf(305.06 KB) Additional Information: full citation, abstract, references, citings, index terms, review

This paper explores and quantifies garbage collection behavior for three whole heap collectors and generational counterparts: *copying semi-space, mark-sweep,* and *reference counting*, the canonical algorithms from which essentially all other collection algorithms are derived. Efficient implementations in MMTk, a Java memory management toolkit, in IBM's Jikes RVM share all common mechanisms to provide a clean experimental platform. Instrumentation separates collector and program behav ...

Keywords: generational, java, mark-sweep, reference counting, semi-space

12 A generational mostly-concurrent garbage collector

Tony Printezis, David Detlefs

October 2000 **ACM SIGPLAN Notices , Proceedings of the 2nd international symposium on Memory management ISMM '00**, Volume 36 Issue 1

Publisher: ACM Press

Full text available: pdf(1.67 MB) Additional Information: full citation, abstract, citings, index terms

This paper reports our experiences with a mostly-concurrent incremental garbage collector, implemented in the context of a high performance virtual machine for the Java™ programming language. The garbage collector is based on the "mostly parallel" collection algorithm of Boehm *et al.* and can be used as the old generation of a generational

memory system. It overloads efficient write-barrier code already generated to support generational garbage collection to also ident ...

13 The treadmill: real-time garbage collection without motion sickness

Henry G. Baker
March 1992 **ACM SIGPLAN Notices**, Volume 27 Issue 3
Publisher: ACM Press
Full text available: pdf(464.25 KB) Additional Information: full citation, abstract, citings, index terms

A simple real-time garbage collection algorithm is presented which does not copy, thereby avoiding some of the problems caused by the asynchronous motion of objects. This in-place "treadmill" garbage collection scheme has approximately the same complexity as other non-moving garbage collectors, thus making it usable in a high-level language implementation where some pointers cannot be traced. The treadmill is currently being used in a Lisp system built in Ada.

14 An on-the-fly reference-counting garbage collector for java

Yossi Levanoni, Erez Petrank
January 2006 **ACM Transactions on Programming Languages and Systems (TOPLAS)**, Volume 28 Issue 1
Publisher: ACM Press
Full text available: pdf(787.15 KB) Additional Information: full citation, abstract, references, index terms

Reference-counting is traditionally considered unsuitable for multiprocessor systems. According to conventional wisdom, the update of reference slots and reference-counts requires atomic or synchronized operations. In this work we demonstrate this is not the case by presenting a novel reference-counting algorithm suitable for a multiprocessor system that does not require any synchronized operation in its write barrier (not even a compare-and-swap type of synchronization). A second novelty of thi ...

Keywords: Programming languages, garbage collection, memory management, reference-counting

15 Mark-copy: fast copying GC with less space overhead

Narendran Sachindran, J. Eliot, B. Moss
October 2003 **ACM SIGPLAN Notices , Proceedings of the 18th annual ACM SIGPLAN conference on Object-oriented programing, systems, languages, and applications OOPSLA '03**, Volume 38 Issue 11
Publisher: ACM Press
Full text available: pdf(297.93 KB) Additional Information: full citation, abstract, references, citings, index terms

Copying garbage collectors have a number of advantages over non-copying collectors, including cheap allocation and avoiding fragmentation. However, in order to provide completeness (the guarantee to reclaim each garbage object eventually), standard copying collectors require space equal to twice the size of the maximum live data for a program. We present a *mark-copy* collection algorithm (MC) that extends generational copying collection and significantly reduces the heap space required to ...

Keywords: Java, copying collector, generational collector, mark-copy, mark-sweep

16 A distributed garbage collector for active objects

Isabelle Puaut
October 1994 **ACM SIGPLAN Notices , Proceedings of the ninth annual conference on Object-oriented programming systems, language, and applications**

OOPSLA '94, Volume 29 Issue 10
Publisher: ACM Press

Full text available: pdf(2.18 MB) Additional Information: full citation, abstract, references, citings, index terms

This paper presents an algorithm that performs garbage collection in distributed systems of active objects (i.e., objects having their own threads of control). Our proposition extends the basic marking algorithm proposed by Kafura in [1] to a distributed environment. The proposed garbage collector is made up of a set of local garbage collectors, one per site, loosely coupled to a (logically centralized) global garbage collector that maintains a global snapshot of the system state relevant t ...

17 The Compressor: concurrent, incremental, and parallel compaction

Haim Kermany, Erez Petrank
June 2006 **ACM SIGPLAN Notices , Proceedings of the 2006 ACM SIGPLAN conference on Programming language design and implementation PLDI '06**, Volume 41 Issue 6
Publisher: ACM Press
Full text available: pdf(483.26 KB) Additional Information: full citation, abstract, references, index terms

The widely used Mark-and-Sweep garbage collector has a drawback in that it does not move objects during collection. As a result, large long-running realistic applications, such as Web application servers, frequently face the fragmentation problem. To eliminate fragmentation, a heap compaction is run periodically. However, compaction typically imposes very long undesirable pauses in the application. While efficient concurrent collectors are ubiquitous in production runtime systems (such as JVMs), ...

Keywords: compaction, concurrent garbage collection, garbage collection, memory management, runtime systems

18 Controlling fragmentation and space consumption in the metronome, a real-time garbage collector for Java

David F. Bacon, Perry Cheng, V. T. Rajan
June 2003 **ACM SIGPLAN Notices , Proceedings of the 2003 ACM SIGPLAN conference on Language, compiler, and tool for embedded systems LCTES '03**, Volume 38 Issue 7
Publisher: ACM Press

Full text available: pdf(354.15 KB) Additional Information: full citation, abstract, references, citings, index terms

Now that the use of garbage collection in languages like Java is becoming widely accepted due to the safety and software engineering benefits it provides, there is significant interest in applying garbage collection to hard real-time systems. Past approaches have generally suffered from one of two major flaws: either they were not provably real-time, or they imposed large space overheads to meet the real-time bounds.Our previous work [3] presented the Metronome, a mostly non-copying real-time co ...

Keywords: compaction, cost model, fragmentation, space bounds

19 Cache performance of garbage-collected programs

Mark B. Reinhold
June 1994 **ACM SIGPLAN Notices , Proceedings of the ACM SIGPLAN 1994 conference on Programming language design and implementation PLDI '94**, Volume 29 Issue 6
Publisher: ACM Press

Full text available: Additional Information: full citation, abstract, references, citings, index

pdf(1.46 MB) terms

As processor speeds continue to improve relative to main-memory access times, cache performance is becoming an increasingly important component of program performance. Prior work on the cache performance of garbage-collected programs either argues or assumes that conventional garbage-collection methods will yield poor performance, and has therefore concentrated on new collection algorithms designed specifically to improve cache-level reference locality. This paper argues to the c ...

20 Robust, distributed references and acyclic garbage collection

Marc Shapiro, Peter Dickman, David Plainfossé

October 1992 **Proceedings of the eleventh annual ACM symposium on Principles of distributed computing PODC '92**

Publisher: ACM Press

Full text available: pdf(1.27 MB) Additional Information: full citation, references, citings, index terms, review

Results 1 - 20 of 200 Result page: **1** 2 3 4 5 6 7 8 9 10 next

Terms used **garbage collection vector bit pointers** Found **42,831** of **186,958**

Sort results by relevance

Display results expanded form

Save results to a Binder

Search Tips

Open results in a new window

Try an Advanced Search

Try this search in The ACM Guide

Results 1 - 20 of 200 Result page: **1** 2 3 4 5 6 7 8 9 10 next

Best 200 shown Relevance scale

1 Static determination of allocation rates to support real-time garbage collection

Tobias Mann, Morgan Deters, Rob LeGrand, Ron K. Cytron

June 2005 **ACM SIGPLAN Notices , Proceedings of the 2005 ACM SIGPLAN/SIGBED conference on Languages, compilers, and tools for embedded systems LCTES '05**, Volume 40 Issue 7

Publisher: ACM Press

Full text available: pdf(175.91 KB) Additional Information: full citation, abstract, references, citings, index terms

While it is generally accepted that garbage-collected languages offer advantages over languages in which objects must be explicitly deallocated, real-time developers are leery of the adverse effects a garbage collector might have on real-time performance. Semiautomatic approaches based on regions have been proposed, but incorrect usage could cause unbounded storage leaks or program failure. Moreover, correct usage cannot be guaranteed at compile time. Recently, real-time garbage collectors have ...

Keywords: allocation rate, real-time garbage collection, static analysis

2 Comparing mark-and sweep and stop-and-copy garbage collection

Benjamin Zorn

May 1990 **Proceedings of the 1990 ACM conference on LISP and functional programming**

Publisher: ACM Press

Full text available: pdf(1.02 MB) Additional Information: full citation, abstract, references, citings, index terms

Stop-and-copy garbage collection has been preferred to mark-and-sweep collection in the last decade because its collection time is proportional to the size of reachable data and not to the memory size. This paper compares the CPU overhead and the memory requirements of the two collection algorithms extended with generations, and finds that mark-and-sweep collection requires at most a small amount of additional CPU overhead (3-6%) but, requires an average of 20% (and up to 40%) less memory t ...

3 Garbage collection for virtual memory computer systems

H. D. Baecker

November 1972 **Communications of the ACM**, Volume 15 Issue 11

Publisher: ACM Press

Full text available: pdf(578.16 KB) Additional Information: full citation, abstract, references, citings

In list processing there is typically a growing demand for space during program execution. This paper examines the practical implications of this growth within a virtual memory computer system, proposes two new garbage collection techniques for virtual memory systems, and compares them with traditional methods by discussion and by simulation.

Keywords: garbage collection, list processing, page tables, paging, segmentation, virtual memory

4 On the usefulness of type and liveness accuracy for garbage collection and leak detection

Martin Hirzel, Amer Diwan, Johannes Henkel

November 2002 **ACM Transactions on Programming Languages and Systems (TOPLAS)**, Volume 24 Issue 6

Publisher: ACM Press

Full text available: pdf(684.85 KB) Additional Information: full citation, abstract, references, index terms

The effectiveness of garbage collectors and leak detectors in identifying dead objects depends on the *accuracy* of their reachability traversal. Accuracy has two orthogonal dimensions: (i) whether the reachability traversal can distinguish between pointers and nonpointers (*type accuracy*), and (ii) whether the reachability traversal can identify memory locations that will be dereferenced in the future (*liveness accuracy*). This article presents an experimental study of the impo ...

Keywords: Conservative garbage collection, leak detection, liveness accuracy, program analysis, type accuracy

5 Cycles to recycle: garbage collection to the IA-64

Richard L. Hudson, J. Elliot Moss, Sreenivas Subramoney, Weldon Washburn

October 2000 **ACM SIGPLAN Notices , Proceedings of the 2nd international symposium on Memory management ISMM '00**, Volume 36 Issue 1

Publisher: ACM Press

Full text available: pdf(1.25 MB) Additional Information: full citation, abstract, citings, index terms

The IA-64, Intel's 64-bit instruction set architecture, exhibits a number of interesting architectural features. Here we consider those features as they relate to supporting garbage collection (GC). We aim to assist GC and compiler implementors by describing how one may exploit features of the IA-64. Along the way, we record some previously unpublished object scanning techniques, and offer novel ones for object allocation (suggesting some simple operating system support that would simplify it ...

6 An efficient parallel heap compaction algorithm

Diab Abuaiadh, Yoav Ossia, Erez Petrank, Uri Silbershtein

October 2004 **ACM SIGPLAN Notices , Proceedings of the 19th annual ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications OOPSLA '04**, Volume 39 Issue 10

Publisher: ACM Press

Full text available: pdf(291.11 KB) Additional Information: full citation, abstract, references, index terms

We propose a heap compaction algorithm appropriate for modern computing environments. Our algorithm is targeted at SMP platforms. It demonstrates high scalability when running in parallel but is also extremely efficient when running single-threaded on a uniprocessor. Instead of using the standard forwarding pointer mechanism for updating pointers to moved objects, the algorithm saves information for a pack of objects. It then does a small computation to process this information and determine ...

Keywords: JVM, compaction, garbage collection, java, parallel compaction, parallel garbage collection

7 Tag-free garbage collection using explicit type parameters

Andrew Tolmach

July 1994 **ACM SIGPLAN Lisp Pointers , Proceedings of the 1994 ACM conference on LISP and functional programming LFP '94**, Volume VII Issue 3

Publisher: ACM Press

Full text available: pdf(1.04 MB) Additional Information: full citation, abstract, references, citings, index terms

We have constructed a practical tag-free garbage collector based on explicit type parameterization of polymorphic functions, for a dialect of ML. The collector relies on type information derived from an explicitly-typed 2nd-order representation of the program, generated by the compiler as a byproduct of ordinary Hindley-Milner type inference. Runtime type manipulations are performed lazily to minimize execution overhead. We present details of our implementation approach, and preliminary per ...

8 On the type accuracy of garbage collection

Martin Hirzel, Amer Diwan

October 2000 **ACM SIGPLAN Notices , Proceedings of the 2nd international symposium on Memory management ISMM '00**, Volume 36 Issue 1

Publisher: ACM Press

Full text available: pdf(1.25 MB) Additional Information: full citation, abstract, citings, index terms

We describe a novel approach to obtaining type-accurate information for garbage collection in a hardware and language independent way. Our approach uses a run-time analysis to propagate pointer/non-pointer information from significant type events (such as allocation, which always returns a pointer). We use this technique to perform a detailed comparison of garbage collectors with different levels of accuracy and explicit deallocation on a range of C programs. We take advantage of the portabil ...

9 MC^2: high-performance garbage collection for memory-constrained environments

Narendran Sachindran, J. Eliot B. Moss, Emery D. Berger

October 2004 **ACM SIGPLAN Notices , Proceedings of the 19th annual ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications OOPSLA '04**, Volume 39 Issue 10

Publisher: ACM Press

Full text available: pdf(503.53 KB) Additional Information: full citation, abstract, references, citings, index terms

Java is becoming an important platform for memory-constrained consumer devices such as PDAs and cellular phones, because it provides safety and portability. Since Java uses garbage collection, efficient garbage collectors that run in constrained memory are essential. Typical collection techniques used on these devices are mark-sweep and mark-compact. Mark-sweep collectors can provide good throughput and pause times but suffer from fragmentation. Mark-compact collectors prevent fragmentation, ...

Keywords: copying collector, generational collector, java, mark-compact, mark-copy, mark-sweep, memory-constrained copying

10 Lock-free garbage collection for multiprocessors

Maurice P. Herlihy, J. E. B. Moss

June 1991 **Proceedings of the third annual ACM symposium on Parallel algorithms**

and architectures
Publisher: ACM Press
Full text available: pdf(801.91 KB) Additional Information: full citation, references, citings, index terms

11 A parallel, incremental, mostly concurrent garbage collector for servers
Katherine Barabash, Ori Ben-Yitzhak, Irit Goft, Elliot K. Kolodner, Victor Leikehman, Yoav Ossia, Avi Owshanko, Erez Petrank
November 2005 **ACM Transactions on Programming Languages and Systems (TOPLAS)**, Volume 27 Issue 6
Publisher: ACM Press
Full text available: pdf(683.50 KB) Additional Information: full citation, abstract, references, index terms

Multithreaded applications with multigigabyte heaps running on modern servers provide new challenges for garbage collection (GC). The challenges for "server-oriented" GC include: ensuring short pause times on a multigigabyte heap while minimizing throughput penalty, good scaling on multiprocessor hardware, and keeping the number of expensive multicycle fence instructions required by weak ordering to a minimum.We designed and implemented a collector facing these demands building on th ...

Keywords: Garbage collection, JVM, concurrent garbage collection

12 Trading data space for reduced time and code space in real-time garbage collection on stock hardware
Rodney A. Brooks
August 1984 **Proceedings of the 1984 ACM Symposium on LISP and functional programming**
Publisher: ACM Press
Full text available: pdf(665.85 KB) Additional Information: full citation, abstract, references, citings, index terms

This paper presents a new storage representation for cons cells (and all other LISP heap data structures) which allows more time efficient LISP with real-time garbage collection on stock hardware. "Stock hardware" refers to common modern architectures for Von Neumann uni-processors (e.g. MC68000, IBM370, VAX, NS32032, etc.). Previous real-time garbage collection schemes have either explicitly required specially tailored hardware in order to avoid multiple order of magnitude slow ...

13 The Compressor: concurrent, incremental, and parallel compaction
Haim Kermany, Erez Petrank
June 2006 **ACM SIGPLAN Notices , Proceedings of the 2006 ACM SIGPLAN conference on Programming language design and implementation PLDI '06**, Volume 41 Issue 6
Publisher: ACM Press
Full text available: pdf(483.26 KB) Additional Information: full citation, abstract, references, index terms

The widely used Mark-and-Sweep garbage collector has a drawback in that it does not move objects during collection. As a result, large long-running realistic applications, such as Web application servers, frequently face the fragmentation problem. To eliminate fragmentation, a heap compaction is run periodically. However, compaction typically imposes very long undesirable pauses in the application. While efficient concurrent collectors are ubiquitous in production runtime systems (such as JVMs), ...

Keywords: compaction, concurrent garbage collection, garbage collection, memory management, runtime systems

14 Memory safety without garbage collection for embedded applications

Dinakar Dhurjati, Sumant Kowshik, Vikram Adve, Chris Lattner

February 2005 **ACM Transactions on Embedded Computing Systems (TECS)**, Volume 4 Issue 1

Publisher: ACM Press

Full text available: pdf(511.25 KB) Additional Information: full citation, abstract, references, index terms

Traditional approaches to enforcing memory safety of programs rely heavily on run-time checks of memory accesses and on garbage collection, both of which are unattractive for embedded applications. The goal of our work is to develop advanced compiler techniques for enforcing memory safety with minimal run-time overheads. In this paper, we describe a set of compiler techniques that, together with minor semantic restrictions on C programs and no new syntax, ensure memory safety and provide most of ...

Keywords: Embedded systems, automatic pool allocation, compilers, programming languages, region management, security, static analysis

15 Myths and realities: the performance impact of garbage collection

Stephen M. Blackburn, Perry Cheng, Kathryn S. McKinley

June 2004 **ACM SIGMETRICS Performance Evaluation Review , Proceedings of the joint international conference on Measurement and modeling of computer systems SIGMETRICS '04/Performance '04**, Volume 32 Issue 1

Publisher: ACM Press

Full text available: pdf(305.06 KB) Additional Information: full citation, abstract, references, citings, index terms, review

This paper explores and quantifies garbage collection behavior for three whole heap collectors and generational counterparts: *copying semi-space, mark-sweep,* and *reference counting*, the canonical algorithms from which essentially all other collection algorithms are derived. Efficient implementations in MMTk, a Java memory management toolkit, in IBM's Jikes RVM share all common mechanisms to provide a clean experimental platform. Instrumentation separates collector and program behav ...

Keywords: generational, java, mark-sweep, reference counting, semi-space

16 List processing in real time on a serial computer

Henry G. Baker

April 1978 **Communications of the ACM**, Volume 21 Issue 4

Publisher: ACM Press

Full text available: pdf(1.55 MB) Additional Information: full citation, abstract, references, citings, index terms

A real-time list processing system is one in which the time required by the elementary list operations (e.g. CONS, CAR, CDR, RPLACA, RPLACD, EQ, and ATOM in LISP) is bounded by a (small) constant. Classical implementations of list processing systems lack this property because allocating a list cell from the heap may cause a garbage collection, which process requires time proportional to the heap size to finish. A real-time list processing system is presented which continuously reclaims garb ...

Keywords: CDR-coding, LISP, compacting, file or database management, garbage collection, list processing, real-time, reference counting, storage allocation, storage management, virtual memory

17 Garbage collection in generic libraries

Gor V. Nishanov, Sibylle Schupp
October 1998 **ACM SIGPLAN Notices , Proceedings of the 1st international symposium on Memory management ISMM '98**, Volume 34 Issue 3
Publisher: ACM Press
Full text available: pdf(1.16 MB) Additional Information: full citation, abstract, references, index terms

This paper demonstrates a unified and garbage-collector independent way to describe the information required for precise collection. Thereby it is possible to construct, a library that can be used with various garbage collectors, without modifying the code of the library or the collector itself. The library design presented applies the adaptor idiom of generic programming which guarantees no overhead incurred if the library is used with manual allocators or with garbage collectors that do not re ...

18 MEDEA'05: Dusty caches for reference counting garbage collection

Scott Friedman, Praveen Krishnamurthy, Roger Chamberlain, Ron K. Cytron, Jason E. Fritts
September 2005 **ACM SIGARCH Computer Archi tecture News , Proceedings of the 2005 workshop on MEmory performance: DEaling with Applications , systems and architecture MEDEA '05**, Volume 34 Issue 1
Publisher: IEEE Computer Society, ACM Press
Full text available: pdf(355.45 KB) Additional Information: full citation, abstract, references, index terms

Reference counting is a garbage-collection technique that maintains a per-object count of the number of pointers to that object. When the count reaches zero, the object must be dead and can be collected. Although it is cannot detect all garbage on its own, it is well suited for some applications and is implemented typically in conjunction with other methods to increase overall precision. A disadvantage of reference counting is the extra storage traffic that is introduced. In this paper, we descr ...

19 Garbage collecting the Internet: a survey of distributed garbage collection

Saleh E. Abdullahi, Graem A. Ringwood
September 1998 **ACM Computing Surveys (CSUR)**, Volume 30 Issue 3
Publisher: ACM Press
Full text available: pdf(337.65 KB) Additional Information: full citation, abstract, references, citings, index terms, review

Internet programming languages such as Java present new challenges to garbage-collection design. The spectrum of garbage-collection schema for linked structures distributed over a network are reviewed here. Distributed garbage collectors are classified first because they evolved from single-address-space collectors. This taxonomy is used as a framework to explore distribution issues: locality of action, communication overhead and indeterministic communication latency.

Keywords: automatic storage reclamation, distributed, distributed file systems, distributed memories, distributed object-oriented management, memory management, network communication, object-oriented databases, reference counting

20 Concurrency: Message analysis-guided allocation and low-pause incremental garbage collection in a concurrent language

Konstantinos Sagonas, Jesper Wilhelmsson
October 2004 **Proceedings of the 4th international symposium on Memory management**
Publisher: ACM Press
Full text available: pdf(650.12 KB) Additional Information: full citation, abstract, references, index terms

We present a memory management scheme for a concurrent programming language

where communication occurs using message-passing with copying semantics. The runtime system is built around process-local heaps, which frees the memory manager from redundant synchronization in a multithreaded implementation and allows the memory reclamation of process-local heaps to be a private business and to often take place without garbage collection. The allocator is guided by a static analysis which speculative ...

Keywords: Erlang, concurrent languages, incremental and real-time garbage collection, thread-local heaps

Terms used **just** in **time** **mark** **sweep** **virtual** **machine**

Found **102,730** of **186,958**

Sort results by: relevance

Display results: expanded form

Save results to a Binder

Search Tips

Open results in a new window

Try an Advanced Search

Try this search in The ACM Guide

Results 1 - 20 of 200 Result page: **1** 2 3 4 5 6 7 8 9 10 next

Best 200 shown

Relevance scale

1 Reducing sweep time for a nearly empty heap

Yoo C. Chung, Soo-Mook Moon, Kemal Ebcioğlu, Dan Sahlin

January 2000 **Proceedings of the 27th ACM SIGPLAN-SIGACT symposium on Principles of programming languages**

Publisher: ACM Press

Full text available: pdf(1.31 MB) Additional Information: full citation, abstract, references, citings, index terms

Mark and sweep garbage collectors are known for using time proportional to the heap size when sweeping memory, since all objects in the heap, regardless of whether they are live or not, must be visited in order to reclaim the memory occupied by dead objects. This paper introduces a sweeping method which traverses only the live objects, so that sweeping can be done in time dependent only on the number of live objects in the heap. This allows each collection to use time independent ...

2 An on-the-fly mark and sweep garbage collector based on sliding views

Hezi Azatchi, Yossi Levanoni, Harel Paz, Erez Petrank

October 2003 **ACM SIGPLAN Notices , Proceedings of the 18th annual ACM SIGPLAN conference on Object-oriented programing, systems, languages, and applications OOPSLA '03**, Volume 38 Issue 11

Publisher: ACM Press

Full text available: pdf(244.12 KB) Additional Information: full citation, abstract, references, citings, index terms

With concurrent and garbage collected languages like Java and C# becoming popular, the need for a suitable non-intrusive, efficient, and concurrent multiprocessor garbage collector has become acute. We propose a novel mark and sweep on-the-fly algorithm based on the sliding views mechanism of Levanoni and Petrank. We have implemented our collector on the Jikes Java Virtual Machine running on a Netfinity multiprocessor and compared it to the concurrent algorithm and to the stop-the-world collecto ...

Keywords: concurrent garbage collection, garbage collection, memory management, on-the-fly garbage collection, runtime systems

3 Very concurrent mark-&-sweep garbage collection without fine-grain synchronization

Lorenz Huelsbergen, Phil Winterbottom

October 1998 **ACM SIGPLAN Notices , Proceedings of the 1st international symposium on Memory management ISMM '98**, Volume 34 Issue 3

Publisher: ACM Press

Full text available: pdf(1.36 MB) Additional Information: full citation, abstract, references, citings, index terms

We describe a new incremental algorithm for the concurrent reclamation of a program's allocated, yet unreachable, data. Our algorithm is a variant of mark-&-sweep collection that---unlike prior designs---runs mutator, marker, and sweeper threads concurrently *without* explicit fine-grain synchronization on shared-memory multiprocessors. A global, but infrequent, synchronization coordinates the per-object coloring marks used by the three threads; fine-grain synchronization is achieve ...

4 Virtual memory on a narrow machine for an object-oriented language

Ted Kaehler

June 1986 **ACM SIGPLAN Notices , Conference proceedings on Object-oriented programming systems, languages and applications OOPLSA '86**, Volume 21 Issue 11

Publisher: ACM Press

Full text available: pdf(1.66 MB) Additional Information: full citation, abstract, references, citings, index terms

LOOM (Large Object-Oriented Memory) is a virtual memory implemented in software that supports the Smalltalk-80(™) programming language and environment on the Xerox Dorado computer. LOOM provides 8 billion bytes of secondary memory address space and is specifically designed to run on computers with a narrow word size (16-bit wide words). All storage is viewed as objects that contain fields. Objects may have an average size as small as 10 fields. LOOM swaps objects between primary and s ...

5 Level set and PDE methods for computer graphics

David Breen, Ron Fedkiw, Ken Museth, Stanley Osher, Guillermo Sapiro, Ross Whitaker

August 2004 **ACM SIGGRAPH 2004 Course Notes SIGGRAPH '04**

Publisher: ACM Press

Full text available: pdf(17.07 MB) Additional Information: full citation, abstract, citings

Level set methods, an important class of partial differential equation (PDE) methods, define dynamic surfaces implicitly as the level set (iso-surface) of a sampled, evolving nD function. The course begins with preparatory material that introduces the concept of using partial differential equations to solve problems in computer graphics, geometric modeling and computer vision. This will include the structure and behavior of several different types of differential equations, e.g. the level set eq ...

6 VSched: Mixing Batch And Interactive Virtual Machines Using Periodic Real-time Scheduling

Bin Lin, Peter A. Dinda

November 2005 **Proceedings of the 2005 ACM/IEEE conference on Supercomputing SC '05**

Publisher: IEEE Computer Society

Full text available: pdf(8.96 MB) Additional Information: full citation, abstract, citings, index terms

We are developing Virtuoso, u system ,for distributed computing using virtual machines (VMs). Virtuoso must be uble to mix batch und interactive VMs on the same physical hardwure, while satisfiing constraint on re- sponsiveness und compute rates for each workload. VSched is the component of Virtuoso that provides this capability. VSched is an entirely user-level tool that interacts with the stock Linux kernel running below any type-11 virtual machine monitor to schedule VMs (indeed, any process) ...

7 Tuning garbage collection for reducing memory system energy in an embedded java environment

G. Chen, R. Shetty, M. Kandemir, N. Vijaykrishnan, M. J. Irwin, M. Wolczko
November 2002 **ACM Transactions on Embedded Computing Systems (TECS)**, Volume 1 Issue 1
Publisher: ACM Press

Full text available: pdf(740.23 KB) Additional Information: full citation, abstract, references, citings, index terms

Java has been widely adopted as one of the software platforms for the seamless integration of diverse computing devices. Over the last year, there has been great momentum in adopting Java technology in devices such as cellphones, PDAs, and pagers where optimizing energy consumption is critical. Since, traditionally, the Java virtual machine (JVM), the cornerstone of Java technology, is tuned for performance, taking into account energy consumption requires reevaluation, and possibly redesign of t ...

Keywords: Garbage collector, Java Virtual Machine (JVM), K Virtual Machine (KVM), low power computing

8 How java programs interact with virtual machines at the microarchitectural level

Lieven Eeckhout, Andy Georges, Koen De Bosschere
October 2003 **ACM SIGPLAN Notices , Proceedings of the 18th annual ACM SIGPLAN conference on Object-oriented programing, systems, languages, and applications OOPSLA '03**, Volume 38 Issue 11
Publisher: ACM Press

Full text available: pdf(348.88 KB) Additional Information: full citation, abstract, references, citings, index terms

Java workloads are becoming increasingly prominent on various platforms ranging from embedded systems, over general-purpose computers to high-end servers. Understanding the implications of all the aspects involved when running Java workloads, is thus extremely important during the design of a system that will run such workloads. In other words, understanding the interaction between the Java application, its input and the virtual machine it runs on, is key to a succesful design. The goal of this ...

Keywords: Java workloads, performance analysis, statistical data analysis, virtual machine technology, workload characterization

9 Effectiveness of cross-platform optimizations for a java just-in-time compiler

Kazuaki Ishizaki, Mikio Takeuchi, Kiyokuni Kawachiya, Toshio Suganuma, Osamu Gohda, Tatsushi Inagaki, Akira Koseki, Kazunori Ogata, Motohiro Kawahito, Toshiaki Yasue, Takeshi Ogasawara, Tamiya Onodera, Hideaki Komatsu, Toshio Nakatani
October 2003 **ACM SIGPLAN Notices , Proceedings of the 18th annual ACM SIGPLAN conference on Object-oriented programing, systems, languages, and applications OOPSLA '03**, Volume 38 Issue 11
Publisher: ACM Press

Full text available: pdf(405.65 KB) Additional Information: full citation, abstract, references, citings, index terms

This paper describes the system overview of our Java Just-In-Time (JIT) compiler, which is the basis for the latest production version of IBM Java JIT compiler that supports a diversity of processor architectures including both 32-bit and 64-bit modes, CISC, RISC, and VLIW architectures. In particular, we focus on the design and evaluation of the cross-platform optimizations that are common across different architectures. We studied the effectiveness of each optimization by selectively disabling ...

Keywords: Java, just-in-time compiler, optimization

10 Software prefetching for mark-sweep garbage collection: hardware analysis and software redesign

Chen-Yong Cher, Antony L. Hosking, T. N. Vijaykumar

October 2004 **ACM SIGOPS Operating Systems Review , ACM SIGPLAN Notices , ACM SIGARCH Computer Architecture News , Proceedings of the 11th international conference on Architectural support for programming languages and operating systems ASPLOS-XI**, Volume 38 , 39 , 32 Issue 5 , 11 , 5

Publisher: ACM Press

Full text available: pdf(165.32 KB) Additional Information: full citation, abstract, references, index terms

Tracing garbage collectors traverse references from live program variables, transitively tracing out the closure of live objects. Memory accesses incurred during tracing are essentially random: a given object may contain references to any other object. Since application heaps are typically much larger than hardware caches, tracing results in many cache misses. Technology trends will make cache misses more important, so tracing is a prime target for prefetching.Simulation of Java benchmarks runni ...

Keywords: breadth-first, buffered prefetch, cache architecture, depth-first, garbage collection, mark-sweep, prefetch-on-grey, prefetching

11 Controlling garbage collection and heap growth to reduce the execution time of Java applications

Tim Brecht, Eshrat Arjomandi, Chang Li, Hang Pham

September 2006 **ACM Transactions on Programming Languages and Systems (TOPLAS)**, Volume 28 Issue 5

Publisher: ACM Press

Full text available: pdf(335.24 KB) Additional Information: full citation, abstract, references, index terms

In systems that support garbage collection, a tension exists between collecting garbage too frequently and not collecting it frequently enough. Garbage collection that occurs too frequently may introduce unnecessary overheads at the risk of not collecting much garbage during each cycle. On the other hand, collecting garbage too infrequently can result in applications that execute with a large amount of virtual memory (i.e., with a large footprint) and suffer from increased execution times due to ...

Keywords: Garbage collection, Java, heap growth, implementation, memory management, performance measurement, programming languages

12 List processing in real time on a serial computer

Henry G. Baker

April 1978 **Communications of the ACM**, Volume 21 Issue 4

Publisher: ACM Press

Full text available: pdf(1.55 MB) Additional Information: full citation, abstract, references, citings, index terms

A real-time list processing system is one in which the time required by the elementary list operations (e.g. CONS, CAR, CDR, RPLACA, RPLACD, EQ, and ATOM in LISP) is bounded by a (small) constant. Classical implementations of list processing systems lack this property because allocating a list cell from the heap may cause a garbage collection, which process requires time proportional to the heap size to finish. A real-time list processing system is presented which continuously reclaims garb ...

Keywords: CDR-coding, LISP, compacting, file or database management, garbage collection, list processing, real-time, reference counting, storage allocation, storage

management, virtual memory

13 Comparing mark-and sweep and stop-and-copy garbage collection

Benjamin Zorn

May 1990 **Proceedings of the 1990 ACM conference on LISP and functional programming**

Publisher: ACM Press

Full text available: pdf(1.02 MB) Additional Information: full citation, abstract, references, citings, index terms

Stop-and-copy garbage collection has been preferred to mark-and-sweep collection in the last decade because its collection time is proportional to the size of reachable data and not to the memory size. This paper compares the CPU overhead and the memory requirements of the two collection algorithms extended with generations, and finds that mark-and-sweep collection requires at most a small amount of additional CPU overhead (3-6%) but, requires an average of 20% (and up to 40%) less memory t ...

14 GPGPU: general purpose computation on graphics hardware

David Luebke, Mark Harris, Jens Krüger, Tim Purcell, Naga Govindaraju, Ian Buck, Cliff Woolley, Aaron Lefohn

August 2004 **ACM SIGGRAPH 2004 Course Notes SIGGRAPH '04**

Publisher: ACM Press

Full text available: pdf(63.03 MB) Additional Information: full citation, abstract, citings

The graphics processor (GPU) on today's commodity video cards has evolved into an extremely powerful and flexible processor. The latest graphics architectures provide tremendous memory bandwidth and computational horsepower, with fully programmable vertex and pixel processing units that support vector operations up to full IEEE floating point precision. High level languages have emerged for graphics hardware, making this computational power accessible. Architecturally, GPUs are highly parallel s ...

15 Older-first garbage collection in practice: evaluation in a Java Virtual Machine

Darko Stefanović, Matthew Hertz, Stephen M. Blackburn, Kathryn S. McKinley, J. Eliot B. Moss

June 2002 **ACM SIGPLAN Notices , Proceedings of the 2002 workshop on Memory system performance MSP '02**, Volume 38 Issue 2 supplement

Publisher: ACM Press

Full text available: pdf(1.15 MB) Additional Information: full citation, abstract, references, citings

Until recently, the best performing copying garbage collectors used a generational policy which repeatedly collects the very youngest objects, copies any survivors to an older space, and then infrequently collects the older space. A previous study that used garbage-collection simulation pointed to potential improvements by using an *Older-First* copying garbage collection algorithm. The Older-First algorithm sweeps a fixed-sized window through the heap from older to younger objects, and avo ...

16 An on-the-fly reference-counting garbage collector for java

Yossi Levanoni, Erez Petrank

January 2006 **ACM Transactions on Programming Languages and Systems (TOPLAS)**, Volume 28 Issue 1

Publisher: ACM Press

Full text available: pdf(787.15 KB) Additional Information: full citation, abstract, references, index terms

Reference-counting is traditionally considered unsuitable for multiprocessor systems. According to conventional wisdom, the update of reference slots and reference-counts requires atomic or synchronized operations. In this work we demonstrate this is not the

case by presenting a novel reference-counting algorithm suitable for a multiprocessor system that does not require any synchronized operation in its write barrier (not even a compare-and-swap type of synchronization). A second novelty of thi ...

Keywords: Programming languages, garbage collection, memory management, reference-counting

17 Concurrent garbage collection using hardware-assisted profiling

Timothy H. Heil, James E. Smith
October 2000 **ACM SIGPLAN Notices , Proceedings of the 2nd international symposium on Memory management ISMM '00**, Volume 36 Issue 1
Publisher: ACM Press
Full text available: pdf(1.74 MB) Additional Information: full citation, abstract, citings, index terms

In the presence of on-chip multithreading, a Virtual Machine (VM) implementation can readily take advantage of *service threads* for enhancing performance by performing tasks such as profile collection and analysis, dynamic optimization, and garbage collection concurrently with program execution. In this context, a hardware-assisted profiling mechanism is proposed. The *Relational Profiling Architecture* (RPA) is designed from the top down. RPA is based on a relational model similar ...

18 New garbage collection algorithms and strategies: Dynamic selection of application-specific garbage collectors

Sunil Soman, Chandra Krintz, David F. Bacon
October 2004 **Proceedings of the 4th international symposium on Memory management**
Publisher: ACM Press
Full text available: pdf(185.74 KB) Additional Information: full citation, abstract, references, index terms

Much prior work has shown that the performance enabled by garbage collection (GC) systems is highly dependent upon the behavior of the application as well as on the available resources. That is, no single GC enables the best performance for all programs and all heap sizes. To address this limitation, we present the design, implementation, and empirical evaluation of a novel Java Virtual Machine (JVM) extension that facilitates dynamic switching between a number of very different and popular g ...

Keywords: Java, annotation, application-specific collection, dynamic selection, hot-swapping, virtual machine

19 Programming languages: Garbage collection for embedded systems

David F. Bacon, Perry Cheng, David Grove
September 2004 **Proceedings of the 4th ACM international conference on Embedded software EMSOFT '04**
Publisher: ACM Press
Full text available: pdf(199.59 KB) Additional Information: full citation, abstract, references, citings, index terms

Security concerns on embedded devices like cellular phones make Java an extremely attractive technology for providing third-party and user-downloadable functionality. However, garbage collectors have typically required several times the maximum live data set size (which is the minimum possible heap size) in order to run well. In addition, the size of the virtual machine (ROM) image and the size of the collector's data structures (metadata) have not been a concern for server- or workstation-orien ...

Keywords: compaction, fragmentation, mark-and-sweep, tracing

20 GCspy: an adaptable heap visualisation framework

Tony Printezis, Richard Jones

November 2002 **ACM SIGPLAN Notices , Proceedings of the 17th ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications OOPSLA '02**, Volume 37 Issue 11

Publisher: ACM Press

Full text available: pdf(215.66 KB) Additional Information: full citation, abstract, references, citings, index terms

GCspy is an architectural framework for the collection, transmission, storage and replay of memory management behaviour. It makes new contributions to the understanding of the dynamic memory behaviour of programming languages (and especially object-oriented languages that make heavy demands on the performance of memory managers). GCspy's architecture allows easy incorporation into *any* memory management system: it is not limited to garbage-collected languages. It requires only small change ...

Keywords: Java, garbage collection, language implementation, memory management, visualisation of objects

Web Results **11** - **20** of about **82,600** for **2002 mark sweep concurrent garbage collection**. (**0.17** seconds)

IBM | Research | David F. Bacon | Publications
The Recycler is a **concurrent** multiprocessor **garbage collector** with extremely ... often exhibits better locality properties than **mark**-and-**sweep** collectors. ...
www.research.ibm.com/people/d/dfb/publications.html - 76k - Cached - Similar pages

[PPT] Engineering a Conservative **Mark-Sweep Garbage Collector**
File Format: Microsoft Powerpoint - View as HTML
Such a **concurrent mark** phase can be "fixed" if we can ... the Usefulness of Type and Liveness Accuracy for **Garbage Collection**", TOPLAS 24, 6, November **2002**. ...
www.research.ibm.com/ismm04/slides/boehm-tutorial.ppt - Similar pages

Who Is Collecting Your Java **Garbage**?
Figure 2 shows **concurrent** GC using marking and sweeping. An example of a **mark-sweep** system is the generational on-the-fly **garbage collector** (T. Domani, ...
doi.ieeecomputersociety.org/10.1109/MITP.2003.1191792 - Similar pages

[PDF] Age-Oriented **Concurrent Garbage Collection**
File Format: PDF/Adobe Acrobat - View as HTML
An on-the-fly **mark**. and **sweep garbage collector** based on sliding view. ... **2002** Conference on Prog. Lang. Design and Impl., pages 153–164, **2002**. ...
www.cs.technion.ac.il/~erez/Papers/ao-cc.pdf - Similar pages

Erez Petrank
A Parallel, Incremental, Mostly **Concurrent Garbage Collection** for Servers. ... techniques to obtain a novel **mark** and **sweep** on-the-fly **garbage collector** with ...
www.cs.technion.ac.il/~erez/projects.html - 32k - Cached - Similar pages

A **concurrent**, generational **garbage collector** for a multithreaded ...
Lorenz Huelsbergen , Phil Winterbottom, Very **concurrent mark-&-sweep garbage collection** without fine-grain synchronization, ACM SIGPLAN Notices, v.34 n.3, ...
www.acm.org/pubs/citations/proceedings/plan/158511/p113-doligez/ - 60k - Cached - Similar pages

[PDF] A Real-Time **Garbage Collector** for Embedded Applications in CLI
File Format: PDF/Adobe Acrobat - View as HTML
conservative **mark-sweep** GC without cooperation of a ... **Concurrent Garbage Collector**. A traditional incremental GC that performs **collection** ...
db.usenix.org/events/vm04/wips/goh.pdf - Similar pages

Richard Jones' **Garbage Collection** Bibliography
Concurrent garbage collection in O2. In M. Jarke, M.J. Carey, K.R. Dittrich, ... Scalable hardware-algorithm for **mark-sweep garbage collection**. ...
www.cs.kent.ac.uk/people/staff/rej/gcbib/gcbibS.html - 59k - Cached - Similar pages

[PDF] Fast **Garbage Collection** without a Long Wait
File Format: PDF/Adobe Acrobat - View as HTML
mark-sweep collector marks live objects, identifies all unmarked. objects, and frees them. ... mostly-**concurrent garbage collector**. In Proceedings of the ...
cs.anu.edu.au/techreports/**2002**/TR-CS-02-06.pdf - Similar pages

Memory Management
We created a new **garbage collection** framework, GCTk, (2001-**2002**) in which we ... It also adds free-list memory managers (e.g., **mark-sweep** and reference ...
www-ali.cs.umass.edu/DaCapo/Memory_Management.html - 32k - Cached - Similar pages

---

Web Results **1** - **10** of about **82,500** for **2002 mark sweep concurrent garbage collection**. (**0.26** seconds)

An On-the-Fly **Mark** and **Sweep Garbage Collector** Based on Sliding ...
11 Very **Concurrent Mark-&-Sweep Garbage Collection** without Fine. ... **2002** 3 Also LFP94 and OOPSLA93 Workshop on Memory Management and Ga. ...
citeseer.ist.psu.edu/645296.html - 28k - Cached - Similar pages

Citations: Very **Concurrent Mark-&-Sweep Garbage Collection** without ...
Very **Concurrent Mark-&-Sweep Garbage Collection** without Fine-Grain Synchronization. In Proceedings of the 1998 International Symposium on Memory Management, ...
citeseer.ist.psu.edu/context/1061789/0 - 23k - Cached - Similar pages
[More results from citeseer.ist.psu.edu]

Welcome to IEEE Xplore 2.0: A multithreaded **concurrent garbage** ...
In this paper, a new multithreaded **concurrent** generational **garbage collector** (MCGC) based on **mark-sweep** with the assistance of reference counting is ...
ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1015550 - Similar pages

[PDF] A multithreaded **concurrent garbage collector** parallelizing the new ...
File Format: PDF/Adobe Acrobat
bage **collector**, very **concurrent mark-sweep garbage** col- ... Arizona, April 3-5, **2002**. [16] K.D. Nilsen and W.J. Schmidt, "A High Performance ...
ieeexplore.ieee.org/iel5/7926/21854/01015550.pdf - Similar pages

Fine-tuning Java **garbage collection** performance
If a Java application suffers from variable pause times due to **garbage collection**, **concurrent mark** can help minimize the pause variation, ...
www.ibm.com/developerworks/library/i-gctroub/ - 65k - Cached - Similar pages

Sensible sanitation -- Understanding the IBM Java **Garbage** ...
verbosegc and a **concurrent mark** System.gc **collection** ... Part 2 reviewed how **garbage collection** works, and covered the three main phases: **mark sweep**, ...
www.ibm.com/developerworks/library/i-**garbage**3.html - 55k - Cached - Similar pages
[More results from www.ibm.com]

Mostly **concurrent** compaction for **mark-sweep** GC
A **Mark-Sweep garbage collector** must occasionally execute a compaction, usually while ... and **concurrent** GC for servers, Proceedings of the ACM SIGPLAN **2002** ...
portal.acm.org/citation.cfm?id=1029877&dl=acm&coll=&CFID=15151515&CFTOKEN=6184618 - Similar pages

A generational mostly-**concurrent garbage collector**
These objects must be rescanned to ensure that the **concurrent** marking phase marks ... An on-the-fly **mark** and **sweep garbage collector** based on sliding views, ...
portal.acm.org/citation.cfm?id=362480&coll=portal&dl=ACM - Similar pages
[More results from portal.acm.org]

[PPT] **Garbage Collection** in Java
File Format: Microsoft Powerpoint - View as HTML
Mark: identify **garbage**; **Sweep**: Find **garbage** on heap, de-allocate it ... **Concurrent Garbage Collection**. -Xconcgc; **Concurrent** GC allows other threads to keep ...
java.quest.com/JUG/meetings/presentations/sep02/JUG%20Sept%20**2002**.PPT -

Similar pages

Space efficient conservative **garbage collection**
13 Detlefs, David L., "**Concurrent Garbage Collection** for C++", ... 14 Daniel R. Edelson, A **mark**-and-**sweep collector** C++, Proceedings of the 19th ACM ...
www.acm.org/pubs/citations/proceedings/pldi/155090/p197-boehm/ - 53k -
Cached - Similar pages

Result Page: **1** 2 3 4 5 6 7 8 9 10 **Next**

2002 mark sweep concurrent garbag Search

Search within results | Language Tools | Search Tips | Dissatisfied? Help us improve

---

Google Home - Advertising Programs - Business Solutions - About Google

RESULT LIST
6 results found in the Worldwide database for:
java and virtual in the title AND **garbage** in the title or abstract
(Results are sorted by date of upload in database)

1 **Leaf avoidance during garbage collection in a Java Virtual Machine**
Inventor: BURKA PETER W (CA); SCIAMPACONE RYAN A (CA); (+2)
Applicant: IBM (US)
EC:
IPC: ***G06F17/30***; **G06F17/30**
Publication info: **US2006074990** - 2006-04-06

2 **Benchmarking Garbage Collection in Java Virtual Machines**
Inventor: LEE WOO-HYONG (KR)
Applicant: SAMSUNG ELECTRONICS CO LTD (KR)
EC:
IPC: ***G06F11/34; G06F9/44; G06F9/45*** (+11)
Publication info: **GB2405506** - 2005-03-02

3 **Method and apparatus for executing multiple JAVA((TM)) applications on a single JAVA((TM)) virtual machine**
Inventor: KIENHOEFER JUERGEN (US); DESHPANDE RANJIT (US)
Applicant: SCO GROUP INC (US)
EC:
IPC: ***G06F9/44; H04L9/00***; **G06F9/44** (+3)
Publication info: **US6931544** - 2005-08-16

4 **Garbage collection in a Java virtual machine**
Inventor: TROTTER MARTIN JOHN (GB)
Applicant: IBM (US)
EC: G06F12/02D2G
IPC: ***G06F12/02***; **G06F12/02**; (IPC1-7): G06F12/02 (+2)
Publication info: **GB2345355** - 2000-07-05

5 **Method for accelerating java virtual machine bytecode verification, just-in-time compilation and garbage collection by using a dedicated co-processor**
Inventor: HENDLER DANNY (IL); LEVY JONATHAN (IL); (+1)
Applicant: NAT SEMICONDUCTOR CORP (US)
EC: G06F9/38S4L; G06F9/445V
IPC: ***G06F9/38; G06F9/445***; **G06F9/38** (+2)
Publication info: **US6473777** - 2002-10-29

6 **Method and apparatus for assisting garbage collection process within a java virtual machine**
Inventor: HUBER GARY DOUGLAS (US); MCCAULEY DONALD WILLIAM (US)
Applicant: IBM (US)
EC: G06F12/02D2G
IPC: ***G06F12/02***; **G06F12/02**; (IPC1-7): G06F17/30
Publication info: **US6070173** - 2000-05-30

RESULT LIST
27 results found in the Worldwide database for:
garbage and collection in the title AND **mark** in the title or abstract
(Results are sorted by date of upload in database)

1 **Leaf avoidance during garbage collection in a Java Virtual Machine**
Inventor: BURKA PETER W (CA); SCIAMPACONE RYAN A (CA); (+2) **Applicant:** IBM (US)
EC: **IPC:** ***G06F17/30***; **G06F17/30**
Publication info: **US2006074990** - 2006-04-06

2 **Efficient parallel bitwise sweep during garbage collection**
Inventor: BLANDY GEOFFREY O (US) **Applicant:** IBM (US)
EC: **IPC:** ***G06F12/14***; **G06F12/14**; (IPC1-7): G06F12/14
Publication info: **US2005278487** - 2005-12-15

3 **Work stealing queues for parallel garbage collection**
Inventor: FLOOD CHRISTINE H (US); DETLEFS DAVID L (US); (+3) **Applicant:** SUN MICROSYSTEMS INC (US)
EC: G06F12/02D2G4G **IPC:** ***G06F9/46; G06F12/02; G06F17/30*** (+4)
Publication info: **US2005132374** - 2005-06-16

4 **Method and system for improving the concurrency and parallelism of mark-sweep-compact garbage collection**
Inventor: SUBRAMONEY SREENIVAS (US); HUDSON RICHARD L (US) **Applicant:**
EC: **IPC:** ***G06F17/30***; **G06F17/30**; (IPC1-7): G06F17/30
Publication info: **US2005198088** - 2005-09-08

5 **Bit vector toggling for concurrent mark-sweep garbage collection**
Inventor: SUBRAMONEY SREENIVAS (US); HUDSON RICHARD (US) **Applicant:**
EC: G06F12/02D2G4 **IPC:** ***G06F12/02; G06F17/30***; **G06F12/02** (+2)
Publication info: **US2005114413** - 2005-05-26

6 **Method and system for the garbage collection of shared data**
Inventor: BORMAN SAMUEL DAVID (GB); TROTTER MARTIN JOHN (GB) **Applicant:** IBM (US)
EC: G06F12/02D2G **IPC:** (IPC1-7): G06F17/30
Publication info: **US2003220952** - 2003-11-27

7 **Trace termination for on-the-fly garbage collection for weakly-consistent computer architecture**
Inventor: KOLODNER ELLIOT K (IL); LEWIS ETHAN (IL); (+1) **Applicant:** IBM (US)
EC: G06F11/34T; G06F12/02D2G4 **IPC:** ***G06F11/34; G06F12/02***; **G06F11/34** (+2)
Publication info: **US2002120823** - 2002-08-29

8 **Method for using cache prefetch feature to improve garbage collection algorithm**
Inventor: SUBRAMONEY SREENIVAS (US); HUDSON RICHARD L (US) **Applicant:**
EC: G06F12/02D2G; G06F12/08B8 **IPC:** ***G06F12/02; G06F12/08***; **G06F12/02** (+2)
Publication info: **US2002199065** - 2002-12-26

9 **WORK-STEALING QUEUES FOR PARALLEL GARBAGE COLLECTION**
Inventor: FLOOD CHRISTINE H; AGESEN OLE; (+3) **Applicant:** SUN MICROSYSTEMS INC (US)
EC: G06F12/02D2G4; G06F12/02D2G4G **IPC:** ***G06F12/02***; **G06F12/02**; (IPC1-7): G06F12/00
Publication info: **WO0188713** - 2001-11-22

10 **Elimination of coloring during object creation for concurrent garbage collection**

Inventor: LEWIS ETHAN (IL)

Applicant: IBM (US)

EC: G06F12/02D2G4

IPC: ***G06F12/02***; **G06F12/02**; (IPC1-7): G06F12/00

Publication info: **US2002147899** - 2002-10-10

RESULT LIST
27 results found in the Worldwide database for:
garbage and collection in the title AND **mark** in the title or abstract
(Results are sorted by date of upload in database)

11 Striding-type generation scanning for parallel garbage collection
Inventor: FLOOD CHRISTINE H (US); DETLEFS DAVID L (US) **Applicant:** SUN MICROSYSTEMS INC (US)
EC: G06F12/02D2G4G **IPC:** ***G06F12/02***; **G06F12/02**; (IPC1-7): G06F17/30
Publication info: **US6526422** - 2003-02-25

12 Local allocation buffers for parallel garbage collection
Inventor: FLOOD CHRISTINE H (US); DETLEFS DAVID L (US); (+1) **Applicant:** SUN MICROSYSTEMS INC (US)
EC: G06F12/02D2G4; G06F12/02D2G4G **IPC:** ***G06F12/02; G06F17/30***; **G06F12/02** (+2)
Publication info: **US6826583** - 2004-11-30

13 Adaptive scheduling of garbage collection in a mobile phone
Inventor: PATEL MARK ARMIN (US) **Applicant:** MOTOROLA INC (US)
EC: G06F12/02D2G **IPC:** ***G06F12/02***; **G06F12/02**; (IPC1-7): G06F12/02
Publication info: **GB2359647** - 2001-08-29

14 Method and system for using a mark-list for garbage collection
Inventor: DUSSUD PATRICK H (US) **Applicant:** MICROSOFT CORP (US)
EC: G06F12/02D2G **IPC:** ***G06F12/02***; **G06F12/02**; (IPC1-7): G06F12/00
Publication info: **US6622226** - 2003-09-16

15 METHOD AND SYSTEM FOR DETECTING AND UNITING IDLE AREAS DURING COLLECTION OF GARBAGE
Inventor: KEAN JEROME KUIPAA **Applicant:** IBM
EC: G06F12/02D2; G06F12/02D2G **IPC:** ***G06F12/00; G06F9/44; G06F12/02*** (+5)
Publication info: **JP2001034532** - 2001-02-09

16 Garbage collection in an object cache
Inventor: MATTIS PETER (US); PLEVYAK JOHN (US); (+4) **Applicant:** INKTOMI CORP (US)
EC: G06F12/02D2; G06F12/02D2G **IPC:** ***G06F12/02***; **G06F12/02**; (IPC1-7): G06F12/00
Publication info: **US6209003** - 2001-03-27

17 METHOD AND DEVICE FOR OPTIMIZING ACCURATE GARBAGE COLLECTION OF ARRAY NODE IN CARD HEAP
Inventor: KNIPPEL ROSS C; BEYLIN BORIS **Applicant:** SUN MICROSYSTEMS INC
EC: G06F12/02D2G4G **IPC:** ***G06F12/00; G06F12/02***; **G06F12/00** (+2)
Publication info: **JP10301837** - 1998-11-13

18 GARBAGE COLLECTION METHOD
Inventor: SHIMURA HIROYA **Applicant:** FUJITSU LTD
EC: **IPC:** ***G06F12/00***; **G06F12/00**; (IPC1-7): G06F12/00
Publication info: **JP11232162** - 1999-08-27

19 GARBAGE COLLECTION EFFICIENCY INCREASING METHOD
Inventor: OTAKE KAZUO **Applicant:** NIPPON ELECTRIC CO
EC: **IPC:** ***G06F12/00***; **G06F12/00**; (IPC1-7): G06F12/00
Publication info: **JP7028691** - 1995-01-31

20 COMPUTER SYSTEM FOR CONSERVATIVE STACK AND GENERATIONAL HEAP-GARBAGE COLLECTION AND METHOD THEREOF
Inventor: JIEEMUZU ERU ADOKOTSUKU **Applicant:** MICROSOFT CORP
EC: G06F12/02D2G; G06F12/02D2G4G **IPC:** ***G06F12/00; G06F12/02***; **G06F12/00** (+2)

Publication info: JP6095954 - 1994-04-08

RESULT LIST
27 results found in the Worldwide database for:
garbage and collection in the title AND **mark** in the title or abstract
(Results are sorted by date of upload in database)

21 GARBAGE COLLECTION METHOD
Inventor: SHINTANI YOSHIHIRO
Applicant: OKI ELECTRIC IND CO LTD
EC:
IPC: ***G06F12/00***; **G06F12/00**; (IPC1-7): G06F12/00
Publication info: **JP4049440** - 1992-02-18

22 MARKING METHOD FOR GARBAGE COLLECTION OF DATA
Inventor: AKIYAMA TOMOKO
Applicant: FUJITSU LTD
EC:
IPC: ***G06F12/00; G06F9/44***; **G06F12/00** (+2)
Publication info: **JP4014152** - 1992-01-20

23 GARBAGE COLLECTION SYSTEM
Inventor: NAKAJIMA KATSUTO; NISHIKAWA HIROSHI; (+1)
Applicant: AGENCY IND SCIENCE TECHN
EC:
IPC: ***G06F12/02; G06F9/44; G06F12/00*** (+5)
Publication info: **JP2022746** - 1990-01-25

24 GARBAGE COLLECTION SYSTEM
Inventor: NAKAJIMA KATSUTO; NISHIKAWA HIROSHI; (+1)
Applicant: AGENCY IND SCIENCE TECHN
EC:
IPC: ***G06F12/02; G06F9/44; G06F12/00*** (+5)
Publication info: **JP2022745** - 1990-01-25

25 COORDINATE TYPE GARBAGE COLLECTION PROCESSING SYSTEM
Inventor: OZAWA TOSHIHIRO
Applicant: FUJITSU LTD
EC:
IPC: ***G06F12/00; G06F9/44; G06F12/02*** (+10)
Publication info: **JP1280849** - 1989-11-13

26 HARDWARE STACK FOR LISP MACHINE IN PARALLEL GARBAGE COLLECTION
Inventor: TERAMURA SHINSUKE; NAKANISHI MASAKAZU
Applicant: RICOH KK
EC:
IPC: ***G06F9/44; G06F12/00; G06F12/02*** (+5)
Publication info: **JP63085947** - 1988-04-16

27 GARBAGE COLLECTION SYSTEM
Inventor: KIMURA YASUNORI
Applicant: FUJITSU LTD
EC:
IPC: ***G06F9/44; G06F12/00; G06F13/00*** (+5)
Publication info: **JP59119459** - 1984-07-10

RESULT LIST
2 results found in the Worldwide database for:
mark and sweep in the title AND **heap** in the title or abstract
(Results are sorted by date of upload in database)

1 **Method and system for improving the concurrency and parallelism of mark-sweep-compact garbage collection**
Inventor: SUBRAMONEY SREENIVAS (US); HUDSON RICHARD L (US)
Applicant:
EC:
IPC: ***G06F17/30***; **G06F17/30**; (IPC1-7): G06F17/30
Publication info: **US2005198088** - 2005-09-08

2 **Bit vector toggling for concurrent mark-sweep garbage collection**
Inventor: SUBRAMONEY SREENIVAS (US); HUDSON RICHARD (US)
Applicant:
EC: G06F12/02D2G4
IPC: ***G06F12/02***; ***G06F17/30***; **G06F12/02** (+2)
Publication info: **US2005114413** - 2005-05-26

RESULT LIST
Approximately **52** results found in the Worldwide database for:
garbage and collection in the title AND **heap** in the title or abstract
(Results are sorted by date of upload in database)

1 **Compact garbage collection tables**
Inventor: TARDITI DAVID R (US) **Applicant:** MICROSOFT CORP (US)
EC: **IPC:** ***G06F17/30***; **G06F17/30**
Publication info: **US7085789** - 2006-08-01

2 **System and method for concurrent compacting self pacing garbage collection using loaded value and access barriers**
Inventor: TENE GIL (US); WOLF MICHAEL A (US) **Applicant:** AZUL SYSTEMS INC (US)
EC: **IPC:** ***G06F17/30***; **G06F17/30**
Publication info: **US2006155791** - 2006-07-13

3 **System and method for performing garbage collection based on unmanaged memory allocations**
Inventor: DUSSUD PATRICK H (US); GEORGE CHRISTOPHER S (US); (+1) **Applicant:** MICROSOFT CORP (US)
EC: **IPC:** ***G06F17/30***; **G06F17/30**
Publication info: **US2006085494** - 2006-04-20

4 **GENERATIONAL GARBAGE COLLECTION METHOD AND GENERATIONAL GARBAGE COLLECTION PROGRAM**
Inventor: KUROMUSHIYA KENICHI **Applicant:** APLIX CORP
EC: **IPC:** ***G06F12/00***; **G06F12/00**
Publication info: **JP2006039877** - 2006-02-09

5 **Free item distribution among multiple free lists during garbage collection for more efficient object allocation**
Inventor: BLANDY GEOFFREY O (US) **Applicant:** IBM (US)
EC: G06F12/02D2; G06F12/02D2G **IPC:** ***G06F12/00***; **G06F12/00**; (IPC1-7): G06F12/00
Publication info: **US2005273568** - 2005-12-08

6 **Assigning sections within a memory heap for efficient garbage collection of large objects**
Inventor: BLANDY GEOFFREY O (US) **Applicant:** IBM (US)
EC: G06F12/02D2G4 **IPC:** ***G06F12/00***; **G06F12/00**; (IPC1-7): G06F12/00
Publication info: **US2005273567** - 2005-12-08

7 **GARBAGE COLLECTION FOR SMART CARDS**
Inventor: TREGER JOERN (DE); PINZINGER ROBERT (DE) **Applicant:** GIESECKE & DEVRIENT GMBH (DE); TREGER JOERN (DE); (+1)
EC: **IPC:** ***G06F12/02***; **G06F12/02**; (IPC1-7): G06F12/02
Publication info: **WO2005093580** - 2005-10-06

8 **Work stealing queues for parallel garbage collection**
Inventor: FLOOD CHRISTINE H (US); DETLEFS DAVID L (US); (+3) **Applicant:** SUN MICROSYSTEMS INC (US)
EC: G06F12/02D2G4G **IPC:** ***G06F9/46; G06F12/02; G06F17/30*** (+4)
Publication info: **US2005132374** - 2005-06-16

9 **Method and system for multiprocessor garbage collection**
Inventor: DUSSUD PATRICK H (US) **Applicant:** MICROSOFT CORP (US)
EC: G06F12/02D2G4 **IPC:** ***G06F12/02***; **G06F12/02**; (IPC1-7): G06F12/00
Publication info: **US2005033781** - 2005-02-10

10 **System and method for performing garbage collection on a large heap**
Inventor: DUSSUD PATRICK H (US) **Applicant:** MICROSOFT CORP

EC: G06F12/02D2G4G IPC: ***G06F12/00; G06F12/02***; **G06F12/00** (+2)

Publication info: **US2005235120** - 2005-10-20

RESULT LIST
Approximately **52** results found in the Worldwide database for:
garbage and collection in the title AND **heap** in the title or abstract
(Results are sorted by date of upload in database)

11 Method and system for improving the concurrency and parallelism of mark-sweep-compact garbage collection
Inventor: SUBRAMONEY SREENIVAS (US); HUDSON RICHARD L (US) **Applicant:**
EC: **IPC:** ***G06F17/30***; **G06F17/30**; (IPC1-7): G06F17/30
Publication info: **US2005198088** - 2005-09-08

12 Bit vector toggling for concurrent mark-sweep garbage collection
Inventor: SUBRAMONEY SREENIVAS (US); HUDSON RICHARD (US) **Applicant:**
EC: G06F12/02D2G4 **IPC:** ***G06F12/02; G06F17/30***; **G06F12/02** (+2)
Publication info: **US2005114413** - 2005-05-26

13 Optimization of memory usage based on garbage collection simulation
Inventor: COHA JOSEPH A (US); KARKARE ASHISH (US); (+1) **Applicant:** HEWLETT PACKARD CO (US)
EC: G06F11/34S **IPC:** ***G06F11/28; G06F9/44; G06F9/46*** (+14)
Publication info: **EP1349077** - 2003-10-01

14 Method and system for the garbage collection of shared data
Inventor: BORMAN SAMUEL DAVID (GB); TROTTER MARTIN JOHN (GB) **Applicant:** IBM (US)
EC: G06F12/02D2G **IPC:** (IPC1-7): G06F17/30
Publication info: **US2003220952** - 2003-11-27

15 Apparatus, method, and program for implementing garbage collection suitable for real-time processing
Inventor: KAWAMOTO TAKUJI (JP) **Applicant:**
EC: G06F12/02D2G4G **IPC:** ***G06F12/00; G06F9/44; G06F9/46*** (+6)
Publication info: **US2003140071** - 2003-07-24

16 Method and apparatus for performing generational garbage collection in a segmented heap
Inventor: NAGARAJAN VIJAY G (US); ROCHETTI ROBERT (US); (+1) **Applicant:**
EC: G06F12/02D2G4G **IPC:** ***G06F12/00; G06F12/02; G06F17/30*** (+4)
Publication info: **US2004003014** - 2004-01-01

17 Garbage collector employing multiple-car collection sets
Inventor: GARTHWAITE ALEXANDER T (US) **Applicant:**
EC: G06F12/02D2G4G **IPC:** ***G06F12/02***; **G06F12/02**; (IPC1-7): G06F12/00
Publication info: **US2002161792** - 2002-10-31

18 Trace termination for on-the-fly garbage collection for weakly-consistent computer architecture
Inventor: KOLODNER ELLIOT K (IL); LEWIS ETHAN (IL); (+1) **Applicant:** IBM (US)
EC: G06F11/34T; G06F12/02D2G4 **IPC:** ***G06F11/34; G06F12/02***; **G06F11/34** (+2)
Publication info: **US2002120823** - 2002-08-29

19 METHODS AND APPARATUS FOR OPTIMIZING GARBAGE COLLECTION
Inventor: WALLMAN DAVID **Applicant:** SUN MICROSYSTEMS INC (US)
EC: G06F9/40; G06F9/42M; (+1) **IPC:** ***G06F9/40; G06F9/42; G06F12/02*** (+3)
Publication info: **WO02054249** - 2002-07-11

20 Computer system with heap reset

Inventor: KOLODNER ELLIOT KARL (IL); LEWIS ETHAN (IL); (+3) **Applicant:** IBM (US)

EC: G06F9/46A2M; G06F12/02D2G4 **IPC:** ***G06F9/50; G06F12/02***; **G06F9/46** (+2)

Publication info: **US2002056019** - 2002-05-09

RESULT LIST
1 result found in the Worldwide database for:
vector in the title AND **garbage** in the title or abstract
(Results are sorted by date of upload in database)

1 **Bit vector toggling for concurrent mark-sweep garbage collection**

Inventor: SUBRAMONEY SREENIVAS (US); HUDSON RICHARD (US)

Applicant:

EC: G06F12/02D2G4

IPC: ***G06F12/02; G06F17/30***; **G06F12/02** (+2)

Publication info: **US2005114413** - 2005-05-26

Data supplied from the ***esp@cenet*** database - Worldwide

RESULT LIST
3 results found in the Worldwide database for:
mark and sweep in the title AND **garbage** in the title or abstract
(Results are sorted by date of upload in database)

1 **Method and system for improving the concurrency and parallelism of mark-sweep-compact garbage collection**
Inventor: SUBRAMONEY SREENIVAS (US); HUDSON RICHARD L (US)
Applicant:
EC:
IPC: ***G06F17/30***; **G06F17/30**; (IPC1-7): G06F17/30
Publication info: **US2005198088** - 2005-09-08

2 **Bit vector toggling for concurrent mark-sweep garbage collection**
Inventor: SUBRAMONEY SREENIVAS (US); HUDSON RICHARD (US)
Applicant:
EC: G06F12/02D2G4
IPC: ***G06F12/02***; ***G06F17/30***; **G06F12/02** (+2)
Publication info: **US2005114413** - 2005-05-26

3 **Adaptive scheduling of garbage collection in a mobile phone**
Inventor: PATEL MARK ARMIN (US)
Applicant: MOTOROLA INC (US)
EC: G06F12/02D2G
IPC: ***G06F12/02***; **G06F12/02**; (IPC1-7): G06F12/02
Publication info: **GB2359647** - 2001-08-29

Results for "(((heap<in>metadata) <and> (garbage<in>metadata))<and> (collection<..." e-mail
Your search matched **18** of **1430374** documents.
A maximum of **100** results are displayed, **25** to a page, sorted by **Relevance** in **Descending** order.

» Search Options

View Session History

New Search

Modify Search

(((heap<in>metadata) <and> (garbage<in>metadata))<and> (collection<in>meta Search

Check to search only within this results set

Display Format: Citation Citation & Abstract

» Key

IEEE JNL	IEEE Journal or Magazine
IEE JNL	IEE Journal or Magazine
IEEE CNF	IEEE Conference Proceeding
IEE CNF	IEE Conference Proceeding
IEEE STD	IEEE Standard

view selected items Select All Deselect All

1. **Hardware support for concurrent garbage collection in SMP systems**
Chang, J.M.; Srisa-An, W.; Chia-Tien Dan Lo;
High Performance Computing in the Asia-Pacific Region, 2000. Proceedings. T International Conference/Exhibition on
Volume 1, 14-17 May 2000 Page(s):513 - 517 vol.1
Digital Object Identifier 10.1109/HPC.2000.846607
AbstractPlus | Full Text: PDF(396 KB) **IEEE CNF**
Rights and Permissions

2. **HeapGuard, eliminating garbage collection in real-time Ada systems**
Harbaugh, S.; Wavering, B.;
Aerospace and Electronics Conference, 1991. NAECON 1991., Proceedings o National
20-24 May 1991 Page(s):704 - 708 vol.2
Digital Object Identifier 10.1109/NAECON.1991.165829
AbstractPlus | Full Text: PDF(360 KB) **IEEE CNF**
Rights and Permissions

3. **Java virtual machine timing probes: a study of object life span and garba**
Qian Yang; Witawas Srisa-an; Skotiniotis, T.; Chang, J.M.;
Performance, Computing, and Communications Conference, 2002. 21st IEEE
3-5 April 2002 Page(s):73 - 80
Digital Object Identifier 10.1109/IPCCC.2002.995138
AbstractPlus | Full Text: PDF(810 KB) **IEEE CNF**
Rights and Permissions

4. **Partitioned garbage collection of a large stable heap**
Liskov, B.; Maheshwari, U.; Ng, T.;
Object-Orientation in Operating Systems, 1996., Proceedings of the Fifth Interr Workshop on
27-28 Oct. 1996 Page(s):117 - 121
Digital Object Identifier 10.1109/IWOOOS.1996.557898
AbstractPlus | Full Text: PDF(568 KB) **IEEE CNF**
Rights and Permissions

5. **Real realtime performance of heap management systems, using increme collectors**
Corporal, H.;

System Sciences, 1991. Proceedings of the Twenty-Fourth Annual Hawaii Inte
Conference on
Volume i, 8-11 Jan. 1991 Page(s):344 - 352 vol.1
Digital Object Identifier 10.1109/HICSS.1991.183904

AbstractPlus | Full Text: PDF(692 KB) **IEEE CNF**
Rights and Permissions

6. **A performance analysis of the active memory system**
Witawas Srisa-An; Srisa-an; Chia-Tien Dan Lo; J Morris Chang;
Computer Design, 2001. ICCD 2001. Proceedings. 2001 International Confere
23-26 Sept. 2001 Page(s):493 - 496
Digital Object Identifier 10.1109/ICCD.2001.955073

AbstractPlus | Full Text: PDF(344 KB) **IEEE CNF**
Rights and Permissions

7. **Evaluation of parallel copying garbage collection on a shared-memory m**
Imai, A.; Tick, E.;
Parallel and Distributed Systems, IEEE Transactions on
Volume 4, Issue 9, Sept. 1993 Page(s):1030 - 1040
Digital Object Identifier 10.1109/71.243529

AbstractPlus | Full Text: PDF(960 KB) **IEEE JNL**
Rights and Permissions

8. **Scalable hardware-algorithm for mark-sweep garbage collection**
Srisa-An, W.; Chia-Tien Dan Lo; Chang, J.M.;
Euromicro Conference, 2000. Proceedings of the 26th
Volume 1, 5-7 Sept. 2000 Page(s):274 - 281 vol.1
Digital Object Identifier 10.1109/EURMIC.2000.874643

AbstractPlus | Full Text: PDF(648 KB) **IEEE CNF**
Rights and Permissions

9. **Do generational schemes improve the garbage collection efficiency?**
Srisa-an, W.; Chang, J.M.; Chia-Tien Dan Lo;
Performance Analysis of Systems and Software, 2000. ISPASS. 2000 IEEE Int
Symposium on
24-25 April 2000 Page(s):58 - 63
Digital Object Identifier 10.1109/ISPASS.2000.842282

AbstractPlus | Full Text: PDF(276 KB) **IEEE CNF**
Rights and Permissions

10. **Automatic heapmanagement and realtime performance**
Corporaal, H.;
CompEuro '91. 'Advanced Computer Technology, Reliable Systems and Appli
Annual European Computer Conference. Proceedings.
13-16 May 1991 Page(s):290 - 295
Digital Object Identifier 10.1109/CMPEUR.1991.257399

AbstractPlus | Full Text: PDF(480 KB) **IEEE CNF**
Rights and Permissions

11. **A shared-memory multiprocessor garbage collector and its evaluation fo choice logic programs**
Imai, A.; Tick, E.;
Parallel and Distributed Processing, 1991. Proceedings of the Third IEEE Sym
2-5 Dec. 1991 Page(s):870 - 877
Digital Object Identifier 10.1109/SPDP.1991.218229

AbstractPlus | Full Text: PDF(716 KB) **IEEE CNF**
Rights and Permissions

12. **On multi-threaded list-processing and garbage collection**

Kuechlin, W.W.; Nevin, N.J.;
Parallel and Distributed Processing, 1991. Proceedings of the Third IEEE Sym
2-5 Dec. 1991 Page(s):894 - 897
Digital Object Identifier 10.1109/SPDP.1991.218226

AbstractPlus | Full Text: PDF(340 KB) **IEEE CNF**
Rights and Permissions

13. Using virtual addresses as object references
Chase, J.; Levy, H.; Tiwary, A.;
Object Orientation in Operating Systems, 1992., Proceedings of the Second In
Workshop on
24-25 Sept. 1992 Page(s):245 - 248
Digital Object Identifier 10.1109/IWOOOS.1992.252974

AbstractPlus | Full Text: PDF(316 KB) **IEEE CNF**
Rights and Permissions

14. Garbage collection of a distributed heap
Ladin, R.; Liskov, B.;
Distributed Computing Systems, 1992., Proceedings of the 12th International C
9-12 June 1992 Page(s):708 - 715
Digital Object Identifier 10.1109/ICDCS.1992.235116

AbstractPlus | Full Text: PDF(820 KB) **IEEE CNF**
Rights and Permissions

15. Architectural support for dynamic memory management
Morris Chang, J.; Srisa-an, W.; Lo, C.-T.D.;
Computer Design, 2000. Proceedings. 2000 International Conference on
17-20 Sept. 2000 Page(s):99 - 104
Digital Object Identifier 10.1109/ICCD.2000.878274

AbstractPlus | Full Text: PDF(492 KB) **IEEE CNF**
Rights and Permissions

16. The generated order preserving real-time garbage collection
Koide, H.;
Real-Time Computing Systems and Applications, 1995. Proceedings., Second
Workshop on
25-27 Oct. 1995 Page(s):168 - 173
Digital Object Identifier 10.1109/RTCSA.1995.528767

AbstractPlus | Full Text: PDF(476 KB) **IEEE CNF**
Rights and Permissions

17. Dynamic memory management for real-time embedded Java chips
Chi-Min Lin; Tien-Fu Chen;
Real-Time Computing Systems and Applications, 2000. Proceedings. Seventh
Conference on
12-14 Dec. 2000 Page(s):49 - 56
Digital Object Identifier 10.1109/RTCSA.2000.896370

AbstractPlus | Full Text: PDF(656 KB) **IEEE CNF**
Rights and Permissions

18. Implementing orthogonal persistence: a simple optimization using replic
Nettles, S.; O'Toole, J.;
Object Orientation in Operating Systems, 1993., Proceedings of the Third Inte
Workshop on
9-10 Dec. 1993 Page(s):177 - 181
Digital Object Identifier 10.1109/IWOOOS.1993.324909

AbstractPlus | Full Text: PDF(336 KB) **IEEE CNF**
Rights and Permissions

Results for "(((garbage<in>metadata) <and> (collection<in>metadata))<and> (vector&..." e-mail
Your search matched **6** of **1430374** documents.
A maximum of **100** results are displayed, **25** to a page, sorted by **Relevance** in **Descending** order.

» Search Options

View Session History

New Search

Modify Search

(((garbage<in>metadata) <and> (collection<in>metadata))<and> (vector<in>met Search

Check to search only within this results set

Display Format: Citation Citation & Abstract

» Key

IEEE JNL	IEEE Journal or Magazine
IEE JNL	IEE Journal or Magazine
IEEE CNF	IEEE Conference Proceeding
IEE CNF	IEE Conference Proceeding
IEEE STD	IEEE Standard

view selected items Select All Deselect All

1. **Comprehensive distributed garbage collection by tracking causal depenc relevant mutator events**
Louboutin, S.R.Y.; Cahill, V.;
Distributed Computing Systems, 1997., Proceedings of the 17th International C
27-30 May 1997 Page(s):516 - 525
Digital Object Identifier 10.1109/ICDCS.1997.603403
AbstractPlus | Full Text: PDF(900 KB) **IEEE CNF**
Rights and Permissions

2. **Consistency maintenance in Web-based real-time group editors**
Chengzheng Sun; Rok Sosic;
Electronic Commerce and Web-based Applications/Middleware, 1999. Proceec International Conference on Distributed Computing Systems Workshops on
31 May-4 June 1999 Page(s):15 - 22
Digital Object Identifier 10.1109/ECMDD.1999.776409
AbstractPlus | Full Text: PDF(168 KB) **IEEE CNF**
Rights and Permissions

3. **Evaluation of an object-caching coprocessor design for object-oriented s**
Chang, J.M.; Gehringer, E.F.;
Computer Design: VLSI in Computers and Processors, 1993. ICCD '93. Proce IEEE International Conference on
3-6 Oct. 1993 Page(s):132 - 139
Digital Object Identifier 10.1109/ICCD.1993.393393
AbstractPlus | Full Text: PDF(572 KB) **IEEE CNF**
Rights and Permissions

4. **Incremental garbage collection for causal relationship computation in dis systems**
Medina, R.;
Parallel and Distributed Processing, 1993. Proceedings of the Fifth IEEE Symp
1-4 Dec. 1993 Page(s):650 - 655
Digital Object Identifier 10.1109/SPDP.1993.395472
AbstractPlus | Full Text: PDF(404 KB) **IEEE CNF**
Rights and Permissions

5. **Geographic information systems and global positioning systems for wate management**

Yan-Guang Chang; Chiou-Hsiung Chen; Hsiu-Lan Huang; Hua-Hung Miao;
Geoscience and Remote Sensing Symposium, 2001. IGARSS '01. IEEE 2001
Volume 5, 9-13 July 2001 Page(s):2112 - 2114 vol.5
Digital Object Identifier 10.1109/IGARSS.2001.977920

AbstractPlus | Full Text: PDF(47 KB) **IEEE CNF**
Rights and Permissions

☐ **6. Exploiting data parallelism for efficient execution of logic programs with bases**
Bansal, A.K.; Potter, J.L.;
Tools for Artificial Intelligence, 1990.,Proceedings of the 2nd International IEEE
6-9 Nov. 1990 Page(s):674 - 681
Digital Object Identifier 10.1109/TAI.1990.130419

AbstractPlus | Full Text: PDF(732 KB) **IEEE CNF**
Rights and Permissions

Results for "(((mark<in>metadata) <and> (sweep<in>metadata))<and> (garbage<in&g..." e-mail
Your search matched **14** of **1430374** documents.
A maximum of **100** results are displayed, **25** to a page, sorted by **Relevance** in **Descending** order.

» Search Options

View Session History

New Search

» Key

IEEE JNL	IEEE Journal or Magazine
IEE JNL	IEE Journal or Magazine
IEEE CNF	IEEE Conference Proceeding
IEE CNF	IEE Conference Proceeding
IEEE STD	IEEE Standard

Modify Search

(((mark<in>metadata) <and> (sweep<in>metadata))<and> (garbage<in>metadat Search

☐ Check to search only within this results set

Display Format: ◉ Citation ○ Citation & Abstract

view selected items Select All Deselect All

☐ **1. Scalable hardware-algorithm for mark-sweep garbage collection**
Srisa-An, W.; Chia-Tien Dan Lo; Chang, J.M.;
Euromicro Conference, 2000. Proceedings of the 26th
Volume 1, 5-7 Sept. 2000 Page(s):274 - 281 vol.1
Digital Object Identifier 10.1109/EURMIC.2000.874643
AbstractPlus | Full Text: PDF(648 KB) **IEEE CNF**
Rights and Permissions

☐ **2. A performance comparison between stop-the-world and multithreaded cc generational garbage collection for Java**
Lo, C.-T.D.; Srisa-an, W.; Chang, J.M.;
Performance, Computing, and Communications Conference, 2002. 21st IEEE
3-5 April 2002 Page(s):301 - 308
Digital Object Identifier 10.1109/IPCCC.2002.995163
AbstractPlus | Full Text: PDF(748 KB) **IEEE CNF**
Rights and Permissions

☐ **3. A multithreaded concurrent garbage collector parallelizing the new instru**
Lo, C.-T.D.; Srisa-an, W.; Chang, J.M.;
Parallel and Distributed Processing Symposium., Proceedings International, IP
Abstracts and CD-ROM
15-19 April 2002 Page(s):59 - 64
Digital Object Identifier 10.1109/IPDPS.2002.1015550
AbstractPlus | Full Text: PDF(213 KB) **IEEE CNF**
Rights and Permissions

☐ **4. Collecting cyclic garbage in distributed systems**
Xinfeng Ye; Keane, J.;
Parallel Architectures, Algorithms, and Networks, 1997. (I-SPAN '97) Proceedi
International Symposium on
18-20 Dec. 1997 Page(s):227 - 231
Digital Object Identifier 10.1109/ISPAN.1997.645100
AbstractPlus | Full Text: PDF(452 KB) **IEEE CNF**
Rights and Permissions

☐ **5. A Scalable Mark-Sweep Garbage Collector on Large-Scale Shared-Memor**
Toshio Endo; Taura, K.; Yonezawa, A.;
Supercomputing, ACM/IEEE 1997 Conference

15-21 Nov. 1997 Page(s):48 - 48
Digital Object Identifier 10.1109/SC.1997.10059

AbstractPlus | Full Text: PDF(160 KB) **IEEE CNF**
Rights and Permissions

☐ **6. Performance enhancements to the Active Memory System**
Witawas Srisa-an; Lo, C.-T.D.; Chang, J.M.;
Computer Design: VLSI in Computers and Processors, 2002. Proceedings. 20
International Conference on
16-18 Sept. 2002 Page(s):249 - 256
Digital Object Identifier 10.1109/ICCD.2002.1106778

AbstractPlus | Full Text: PDF(2149 KB) **IEEE CNF**
Rights and Permissions

☐ **7. Predicting scalability of parallel garbage collectors on shared memory m**
Endo, T.; Taura, K.; Yonezawa, A.;
Parallel and Distributed Processing Symposium., Proceedings 15th Internation
23-27 April 2001 Page(s):6 pp.
Digital Object Identifier 10.1109/IPDPS.2001.924980

AbstractPlus | Full Text: PDF(200 KB) **IEEE CNF**
Rights and Permissions

☐ **8. LaTTe: a Java VM just-in-time compiler with fast and efficient register allc**
Byung-Sun Yang; Soo-Mook Moon; Seongbae Park; Junpyo Lee; SeungIl Lee;
Chung, Y.C.; Suhyun Kim; Ebcioglu, K.; Altman, E.;
Parallel Architectures and Compilation Techniques, 1999. Proceedings. 1999 I
Conference on
12-16 Oct. 1999 Page(s):128 - 138
Digital Object Identifier 10.1109/PACT.1999.807503

AbstractPlus | Full Text: PDF(420 KB) **IEEE CNF**
Rights and Permissions

☐ **9. Reliable garbage collection in distributed object oriented systems**
Gupta, A.; Fuchs, W.K.;
Computer Software and Applications Conference, 1988. COMPSAC 88. Proce
International
5-7 Oct. 1988 Page(s):324 - 328
Digital Object Identifier 10.1109/CMPSAC.1988.17194

AbstractPlus | Full Text: PDF(416 KB) **IEEE CNF**
Rights and Permissions

☐ **10. Distributed EZ [string processing language]**
Campos, A.E.; Hanson, D.R.;
Computer Software and Applications Conference, 1992. COMPSAC '92. Proce
Sixteenth Annual International
21-25 Sept. 1992 Page(s):136 - 142
Digital Object Identifier 10.1109/CMPSAC.1992.217590

AbstractPlus | Full Text: PDF(592 KB) **IEEE CNF**
Rights and Permissions

☐ **11. Java virtual machine timing probes: a study of object life span and garba**
Qian Yang; Witawas Srisa-an; Skotiniotis, T.; Chang, J.M.;
Performance, Computing, and Communications Conference, 2002. 21st IEEE
3-5 April 2002 Page(s):73 - 80
Digital Object Identifier 10.1109/IPCCC.2002.995138

AbstractPlus | Full Text: PDF(810 KB) **IEEE CNF**
Rights and Permissions

12. Hardware support for concurrent garbage collection in SMP systems

Chang, J.M.; Srisa-An, W.; Chia-Tien Dan Lo;
High Performance Computing in the Asia-Pacific Region, 2000. Proceedings. International Conference/Exhibition on
Volume 1, 14-17 May 2000 Page(s):513 - 517 vol.1
Digital Object Identifier 10.1109/HPC.2000.846607

AbstractPlus | Full Text: PDF(396 KB) **IEEE CNF**
Rights and Permissions

13. Do generational schemes improve the garbage collection efficiency?
Srisa-an, W.; Chang, J.M.; Chia-Tien Dan Lo;
Performance Analysis of Systems and Software, 2000. ISPASS. 2000 IEEE In Symposium on
24-25 April 2000 Page(s):58 - 63
Digital Object Identifier 10.1109/ISPASS.2000.842282

AbstractPlus | Full Text: PDF(276 KB) **IEEE CNF**
Rights and Permissions

14. Practical distributed garbage collection for networks with asynchronous message delay
Goug Kwan; Chin, F.;
Parallel and Distributed Systems, 1994. International Conference on
19-21 Dec. 1994 Page(s):406 - 411
Digital Object Identifier 10.1109/ICPADS.1994.590347

AbstractPlus | Full Text: PDF(532 KB) **IEEE CNF**
Rights and Permissions

Results for "(((live<in>metadata) <and> (object<in>metadata))<and> (heap<in>..." e-mail
Your search matched **2** of **1430374** documents.
A maximum of **100** results are displayed, **25** to a page, sorted by **Relevance** in **Descending** order.

» Search Options

View Session History

New Search

Modify Search

(((live<in>metadata) <and> (object<in>metadata))<and> (heap<in>metadata)) Search

☐ Check to search only within this results set

Display Format: ◉ Citation ○ Citation & Abstract

» Key

IEEE JNL	IEEE Journal or Magazine
IEE JNL	IEE Journal or Magazine
IEEE CNF	IEEE Conference Proceeding
IEE CNF	IEE Conference Proceeding
IEEE STD	IEEE Standard

view selected items Select All Deselect All

☐ **1. A performance analysis of the active memory system**
Witawas Srisa-An; Srisa-an; Chia-Tien Dan Lo; J Morris Chang;
Computer Design, 2001. ICCD 2001. Proceedings. 2001 International Confere
23-26 Sept. 2001 Page(s):493 - 496
Digital Object Identifier 10.1109/ICCD.2001.955073
AbstractPlus | Full Text: PDF(344 KB) **IEEE CNF**
Rights and Permissions

☐ **2. A high-performance memory allocator for memory intensive applications**
Chang, J.M.; Hasan, Y.; Lee, W.H.;
High Performance Computing in the Asia-Pacific Region, 2000. Proceedings. 1
International Conference/Exhibition on
Volume 1, 14-17 May 2000 Page(s):6 - 12 vol.1
Digital Object Identifier 10.1109/HPC.2000.846507
AbstractPlus | Full Text: PDF(528 KB) **IEEE CNF**
Rights and Permissions

Wed, 1 Nov 2006, 6:13:16 PM EST

Search Query Display

Edit an existing query or compose a new query in the Search Query Display.

Select a search number (#) to:

- Add a query to the Search Query Display
- Combine search queries using AND, OR, or NOT
- Delete a search
- Run a search

Recent Search Queries

#1 ((mark<in>metadata)) <and> (pyr >= 1950 <and> pyr <= 2002)

#2 (((mark<in>metadata) <and> (sweep<in>metadata))<and> (garbage<in>metadata)) <and> (pyr >= 1950 <and> pyr <= 2002)

#3 (((mark<in>metadata) <and> (sweep<in>metadata))<and> (garbage<in>metadata)) <and> (pyr >= 1950 <and> pyr <= 2002)

#4 (((mark<in>metadata) <and> (sweep<in>metadata))<and> (garbage<in>metadata)) <and> (pyr >= 1950 <and> pyr <= 2002)

#5 (((mark<in>metadata) <and> (sweep<in>metadata))<and> (java<in>metadata)) <and> (pyr >= 1950 <and> pyr <= 2002)

#6 (((mark<in>metadata) <and> (sweep<in>metadata))<and> (java<in>metadata)) <and> (pyr >= 1950 <and> pyr <= 2002)

#7 (((mark<in>metadata) <and> (sweep<in>metadata))<and> (java<in>metadata)) <and> (pyr >= 1950 <and> pyr <= 2002)

#8 (((mark<in>metadata) <and> (sweep<in>metadata))<and> (vector<in>metadata)) <and> (pyr >= 1950 <and> pyr <= 2002)

#9 (((garbage<in>metadata) <and> (sweep<in>metadata)) <and> (vector<in>metadata)) <and> (pyr >= 1950 <and> pyr <= 2002)

#10 (((garbage<in>metadata) <and> (collection<in>metadata)) <and> (vector<in>metadata)) <and> (pyr >= 1950 <and> pyr <= 2002)

#11 (((vector<in>metadata) <and> (thread<in>metadata))<and> (garbage<in>metadata)) <and> (pyr >= 1950 <and> pyr <= 2002)

#12 (((vector<in>metadata) <and> (thread<in>metadata))<and> (sweep<in>metadata)) <and> (pyr >= 1950 <and> pyr <= 2002)

(((vector<in>metadata) <and> (thread<in>metadata))<and>

#13	(pointer<in>metadata)) <and> (pyr >= 1950 <and> pyr <= 2002)
#14	(((heap<in>metadata) <and> (garbage<in>metadata))<and> (collection<in>metadata)) <and> (pyr >= 1950 <and> pyr <= 2002)
#15	(((heap<in>metadata) <and> (garbage<in>metadata))<and> (collection<in>metadata)) <and> (pyr >= 1950 <and> pyr <= 2002)
#16	(((heap<in>metadata) <and> (garbage<in>metadata))<and> (collection<in>metadata)) <and> (pyr >= 1950 <and> pyr <= 2002)
#17	(((heap<in>metadata) <and> (garbage<in>metadata))<and> (collection<in>metadata)) <and> (pyr >= 1950 <and> pyr <= 2002)
#18	(((live<in>metadata) <and> (object<in>metadata))<and> (heap<in>metadata)) <and> (pyr >= 1950 <and> pyr <= 2002)
#19	(((live<in>metadata) <and> (object<in>metadata))<and> (heap<in>metadata)) <and> (pyr >= 1950 <and> pyr <= 2002)
#20	(((live<in>metadata) <and> (object<in>metadata))<and> (heap<in>metadata)) <and> (pyr >= 1950 <and> pyr <= 2002)
#21	(((live<in>metadata) <and> (object<in>metadata))<and> (heap<in>metadata)) <and> (pyr >= 1950 <and> pyr <= 2002)